



## もくじ

### はじめに

第1章	コンピュータの使い方	1
	接続のしかた	1
	SR-1300、SC-1000をお持ちの方	3
	SC-2000をお持ちの方	1
	キーボードの使い方	2
	ケーブル	2
	キーのこと(文字と記号のキー、特殊キー)	4
第2章	メニュー紹介	17
	スタートゲーム	18
	けいさんボード	26
	BASIC	29
	パソコンのへんこう	29

第3章 BASIC	38
ダイレクトモード (直接命令)	38
文字や記号を書く	38
簡単な計算	40
プログラム・カート その1	43
あなたの名前をコンピュータが書きます	43
(PRINT, GOTO, FOR ~ NEXT ~ STEP, NEW, END, RUN, CLS, STOP, CONT)	
プログラムは保存できます	52
(SAVE, VERIFY, LOAD)	
知っているると便利です	55
(LIST, AUTO, REM, CURSOR, SPC, TAB, 記号各種, PRE, CONSOLE)	
プログラム・カート その2	67
もっといろいろな計算できます	67
(LET, INPUT, 変数)	
プログラム・カート その3	72
合格と不合格にふり分けます (IF ~ THEN)	72
オチウケモノは ? (ON ~ GOTO)	74
デジタリ時計に早がわり ? (TIME\$)	78
たくさんさんのデータも (READ, DATA)	79

配列 (もうひとつの変数)	85
一次元配列	86
二次元配列	86
配列のとりけし (ERASE)	89
文字列関数と関数	90
文字列と数値の入力変え (VAL, STR \$)	90
アスキーコードのこと (ASC, CHR \$)	91
文字列を扱う (LEN, LEFT \$, RIGHT \$, MID \$)	93
数値を扱う (SGN, ABS, HEX \$)	95
音をブレイしよう	97
PLAY/PLEN/SOUND/BEEP	
グラフィックス	120
SCREEN/LINE/CIRCLE/PSET/PRINT/	
CURSOR/COLOR/PAINT/SPRITE/PATTERN/	
MAG/SCON/SAVE V/VERIFY V/LOAD V/	
RND/INKEY \$/SYSTEM/VPOKE/VPEEK	
プログラムを制御するには	
(オーバーフローになったとき)	182

③。イスタックを使って「おまかせ」イスタックをお持ちの方	165
STRIG/STCK	
少し高度なことを知りたい人	168
CALL/POKE/LIMIT/PEEK/OUT/INP	
④。ゲームプログラム	172
付録	
変数・配列・定数	188
コントロールコード	191
キャラクターコード	193
BASIC の SET PATTERN とパターンテーブルとの関係	196
メインメモリアップ、VRAM アップ	198
エラーメッセージ（アルファベット順）	200
コマンド・ステートメント・関数・索引	202

## はじめに

セガのホームページをお買い上げいただきましてありがとうございます。このホームページには、ページと、プログラムを作らなくても楽しめる3つのプログラムが用意されています。

ページでプログラムを作るのもよし、基本的な「ゲーム」で遊ぶのもよし、「バスターンのへんこう」でキャラクターをつくり、「ゲーム」に登場させるもよし、「計算ボード」で計算するもよし……。使い方はあなた次第、自由にこのホームページを使いましょう。

エラーが出ても、おそれずにキーボードに触れてください。きっとパソコンはあなたの友だちになってくれることでしょう。ホームページは、パソコンを始める人が、楽しみながら覚えられるようにと作られたものなのですから。

## このホームページは

### ● ひらがなが使えます。

SC-3000 ホームページで、「ひらがな」は使えませんでしたが、でも、ホームページがウェブなら、カタカナだけでなく、ひらがなも使えます。

### ● 文字が大きくなりました。

SC-3000 ホームページより文字が大きくなり、読みやすくなりました。

### ● グラフィックスが強化されました。

描いたものをテープに保存できます。ですから、属に入った絵をいつでも取り出して使うことができます。

### ● 画像の調整が簡単になりました。

「」と自筆を置くだけで画像が変えられます

### ● 変換型になりました。

変換型ホームページとは、小容量を持っていないホームページのことです。変換型にしたことで、ホームページの処理がはやくなり、ゲームなど作るのに有利になりました。

- グラフィックス記号が少なくなりました。

SC-3000 や SK-1000 のキーボードに添かれた **GRAPH** (グラフィックス) キー記号の数は、不届なものに過ぎました。(キーの使い方をご覧ください。)

- **FUNC** キーの働きがありません。

SC-3000 や SK-1000 のキーボードに添かれた **FUNC** キーの命令は **MODE** キー記号では使えません。

これは SC-3000 ベーザックで使います。

## 2つのベーザックの使用目的 ——

SC-3000 ベーザック

本格的プログラミング

精度の高い計算用

ホームベーザック

計算精度は必要としないが、グラフィックスや  
音源を簡単に使いたい

ベーザックの基本を知りたい

目的にあわせてお使いください。





SC-3000 シリーズ SK-1100 専用

# ホームベーシック

テキスト

SEGA

## 第1章 コンピュータの使い方

### 接続のしかた

#### SK-1100、SG-1000をお持ちの方

1. SG-1000 を増着して下さい。  
(電源はさし入れないで下さい。)
2. SK-1100 本体より出ているケーブルを、SG-1000 の裏に付いているケーブルにしっかりと差し込んでください。  
コネクタを差し込むとき、力を入れすぎないように注意してください。
3. 接続が終わったら、HOME BASIC カートリッジを差し込んで、SG-1000 の電源を入れてください。

#### SK-1100 と SG-1000

SG-1000 にはキーボードがありませんので、パソコンとして使えませんので、SK-1100 キーボードを専用 BASIC と共に用いることで、高性能パソコンとして使っていたようになります。

#### SG-3000 をお持ちの方

1. ディスケットを、アタリに接続してください。  
(セリフ入力のあるテレビの場合は、専用ケーブルで、本体とテレビの入力端子と音声端子に接続してください。)

- ② 本体と電源ケーブルを COMPUTER (コンピュータ) 側につなぎ、本体のチャンネルを1または2チャンネルから空いている方にしてください。
- ③ 本体のチャンネル切り替えスイッチを、1または2チャンネルに合わせてください。
- ④ ケーブルコネクタケーブルを正しく差し込んでください。
- ⑤ 接続が終わったら、ケーブルの接続を確認してから、テレビの電源を入れてください。
- ⑥ コンピュータの電源も入れてください。

## キーボードの使い方

### カーソル


画面左上に  が内蔵しています。これは、カーソルといって、キーから入力した文字や記号が表示される位置を示すものです。

カーソルには、    の4種類があり、

		英数モード
		カナモード
		ひらがなモード
		グラフモード

であることを示します。

ふつう、カーソルは ■ の状態です。  キー、或いは **GRAPH** キーを押すことによって、カーソルを変え、キーから入力するものを、カタ、ひらがな、あるいは記号というように、選択することができます。

 キー（キーボード左下にあります。）を押してください。カーソルに ■ になります。同じキーをもう一度押してください。 + になります。さらにもう一度押すと、もとのカーソル ■ にもどります。 **GRAPH** キー（  キーのとなりにあります。）を押すと、 **半** になります。

何か入力するときは、必ず適切なモードを選んでください。

## キーのこと

キーボードには、文字や記号の入力キーと、ほかの用途で使うの鍵もいろいろあります。

（図 2-1-1）










これらのキーは文字や記号の入力キー（たとえば ）と、それ以外の特殊なキー（ など）に分類されます。

### \*\*\* 文字と記号のキーのこと \*\*\*

キーボードには、英文字、数字、カナ文字、記号が3～4種類書き込まれたキーがあります。それらを使うと、文字や記号を両面に表示することができます。

まず、 キー、或いは **GRAPH** キーを使って、適切なモードを選びます。そうして、文字や記号のキーを単独で押す、或いは **SHIFT** キーという特殊キーと共に押さえると、お望みの文字や記号が両面に表示されます。

両面に表示したいもの	モード (カーソル)	キーの押しかた
英文字 (大文字) 数字		英文字や数字のキーを打ちます。
英文字 (小文字)		<b>SHIFT</b> キーを押しながら、英文字のキーを打ちます。
ひらがな		カナ文字のキーを打ちます。
ひらがな ・小文字 (あいうえおかきくけこ) ・を		<b>SHIFT</b> キーを押しながら、カナ文字のキーを打ちます。
カタカナ		カナ文字のキーを打ちます。
カタカナ ・小文字 (アイウエオカキクケコ) ・を		<b>SHIFT</b> キーを押しながら、カナ文字のキーを打ちます。
グラフ・ョク記号		グラフ・ョク記号のキーを打ちます。
グラフ・ョク記号		<b>SHIFT</b> キーを押しながら、グラフ・ョクキーを打ちます。

英数キー： 図

(図に記した記号) 及び英大文字、数字





英語モードで **SHIFT** キーを押す

画面上に出た記号、及び英小文字



# ■ 点検項目

点検項目、或いはその点検項目、半角点、長方形符号及びその点検項目、点検項目



もしも「■」で **SHIFT** キーを押す

図に記した記号、及びオキナウの付いた文字







\*\*\* 特殊キーのこと \*\*\*

ここでは、**[SHIFT]** **[CTRL]** の2つの特殊な機能を説明します。



## ④ 文字を消す・文字の訂正

### **HOME/CLM** (ホーム/クリア)

このキーを押すと、画面上の文字が消えて、カーソルは左上のホームポジションに戻ります。画面  
上の文字、記号を消したいときに使ってください。また、キーを打ち始める前に使ってください。

note / **SHIFT** キーを押しながら、このキーを押すと、文字は消えずにカーソルだけがホ  
ームポジションに戻ります。

### **INS/DEL** (インサート/デリート)

#### デリート

このキーは、文字を一文字ずつ消したり追加する場合に使います。たとえば、A B C D と入力す  
る所を、A B C E と打ってしまった場合、**INS/DEL** キーを押すと、カーソルが一文字分、前  
にもどってBの文字を消します。そこにDの文字を入力しますと、A B C D というふうに、訂正さ  
れます。

A B C E ■ ← Dのかわりに、間違ってEと打ってしまいました。

A B C ■ ← **INS/DEL** キーを押しますと、カーソルが左にひとつ移動  
します。

A B C D ■ ← Dの文字を打ちます。訂正されました。

## インサート

このキーと **[SHIFT]** キーを合わせて用いると、カーソルの点滅が「挿入点滅、インサートモード」という状態になります。たとえば、A B C D の B と C の間に T という文字を入れる場合、次のようになります。

A B C D ■

A B ■ D ← カーソルをCの上に移動させ、**[SHIFT]** キーと **[INS/DEL]** キーを押します。(インサートモードになりました。)

A B T ■ D ← T の文字を入力します。

A B T C D ■ ← 入力した文字が入り、C D が右側にずれました。

インサートモードから抜け出すには

- ① **[CR]** キーを押す。
- ② カーソルキーを押す。
- ③ **[SHIFT]** + **[INS/DEL]** キーを押す。

のいずれかの方法をとってください。

*note* / **INS** (インサート) は文字を追加する意味です。

**DEL** (デリート) は、文字を消す意味です。

＜入力したものを記憶させる＞

**[CR]** (キャリッジ・リターン又はリターン) キーで画面に命令文を入力しても、まだ命令を実



行しませんし、メモリーに記憶もされません。命令文を実行させた後、メモリーに記憶させるためには、**[CR]** キーを押します。プログラムを修正した場合も、**[CR]** キーを押します。

## ＜その他の特殊キー＞

### **[ ] / BREAK** (画面切替/ブレーク)

このキーは二つの働きがあります。**[ ]** (画面の切り替え) と **BREAK** (ブレーク) です。

**[ ]** ——— 画面を切替えるときに使います。コンピュータは、テキスト画面 (プログラムを書く画面) と、グラフィック画面 (グラフィックスを表示する画面) を持っています。その画面の切替えに使います。

詳しくはグラフィックスの **SCREEN** のところを見てください。

**BREAK** ——— プログラムの実行中、プログラムを停止させたい時に使います。

### **[RESET]**

電源を入れた時の画面 (オーブニング画面) と同じ状態に戻すキーです。プログラムの実行中、画面に異常が出た場合、このキーを押すと 1 ～ 2 秒後に、オーブニングの画面にもどります。

### **[SPACE]** (スペースキー)

文字や記号のあいだをあげます。



スペースキーを 1 回押すと、一文字分の空間ができます。



カーソルを上下左右に移動させます。

**GRAPH** (グラフモードキー)

**MODE** (モードキー)

どちらも、モードを選択するのに使います。これらのキーを押すと、モードの形が変わります。

**SHIFT** (シフトキー)

他のキーとあわせて使うことで、さまざまな機能を発揮します。キーの押し方の表を見てください。

**CTRL** (コントロールキー)

このキーは、他の文字キーとあわせて使います。たとえば、**CTRL** キーを押したまま **H** の文字キーを打ちますと、文字を消す動作をします。

付録の「コントロールコード」をみてください。

## 第2章 メニュー紹介

パソコン本体の接続は止しくできましたか。

電源を入れると、宇宙船の機内室があらわれて、そこにメニューがります

1. シュートゲーム
2. けいせんオート
3. BASIC
4. パターンのへんこう

このゲームメニューには、BASICのほか、3つのプログラムが組み込まれています  
あなたはどれを選びますか？

好きなプログラムの番号キーを押しましょう。

## 1. シュートゲーム

1のキーを押すと、シュートゲームの画面になります。

### ＜プレイ方法＞

ジョイスティックでもキーボードでもプレイできます。（ただし、キーボードでは、右側のプレイジョイスティックが動きません。）

	ジョイスティック	キーボード
スタートのしかた	ジョイスティックを2つ同時に押す。	HOME/CLRキーとINS/DELキーを同時に押す。
プレイマシンの 操縦	ジョイスティックの レバー	カーソルキー
ミサイルの発射	ジョイスティック	HOME/CLRキーもしくはINS/ DELキー

得点は、すべて 10 点です。

このゲームをもとにして、いろいろと変えることができます。

#### ＜へんこうできるもの＞

- ① キャラクタ（プレイヤーや敵）の形      メニュー の 4（パターン）のへんこうで読取します。
- ② キャラクタの色      }
- ③ キャラクタの動き      } 「ジョーゲームの内容変更」で説明します。
- ④ ゲームの背景      BASIC のグラフィックスで描いた絵は、自動的に、ジョーゲームの背景になります。  
（ただし、電源を切ると消えてしまいます。）

## シュートゲームの内容変更

BASICからシュートゲームの内容をPOKE文を使って変更することが出来ます。

### プレイヤジョブのキャラクタ変更

プレイヤジョブのキャラクタは、1Pが00～7まで、2Pが8～15まで、計16個用意されています。電源を入れたとき、プレイヤジョブの進む方向によってキャラクタ番号が切り替わっています。切り替えを中止させることもできます。

#### キャラクタ番号と方向の関係

キャラクタ番号		進む方向
1P	2P	
0	8	＼ ↑ ／
1	9	／ ↓ ＼
2	10	
3	11	
4	12	←
5	13	
6	14	→
7	15	

メニューからBASICに入ります。

直接命令かリストで式を実行します。

○キャラクタ切り替えしない ( )は10進数

POKE &H8307, &H00 (0)

○キャラクタ切り替えする

POKE &H8307, &H01 (1)

# ◀プレイヤ数の変更▶

二人用にする

POKE &H8306, &H00 (0)

一人用にする

POKE &H8308, &H01 (1)

10進数

# ◀遠射する弾数の変更▶

一発

POKE &H8309, &H01 (1)

二発

POKE &H8309, &H02 (2)

三発

POKE &H8309, &H03 (3)

# ◀弾の発射方向の変更▶

発射方向 上方のみ

POKE &H830F, &H00 (0)

進行方向

POKE &H830F, &H01 (1)

# ◀プレイヤーの持数▶

持数の設定

POKE &H8313, &H01 ~ &H07 (1 ~ 7)

7機以上設定しても7機しか表示されません。

## ⑤ フルーツ数の変更

10 進数

一画面のフルーツ数

POKE &H830E, &H01 ~ &H08 (1 ~ 8)

## ⑥ フルーツの場所

左側のフルーツを 1 番として

1 番のフルーツ	X 座標	&H8320, &H00 ~ &HFF	(0 ~ 255)
	Y 座標	&H8321, &H00 ~ &H8F	(0 ~ 191)
↓ (アドレスの変化幅は 2 です。)			
8 番のフルーツ	X 座標	&H832E, &H00 ~ &HFF	(0 ~ 255)
	Y 座標	&H832F, &H00 ~ &H8F	(0 ~ 191)

## ⑦ フルーツの色を変える

最初画面を 1 画として

1 画のフルーツ色番号	&H8330, &H01 ~ &H0F	(0 ~ 15)
使用するパターン番号	&H8331, &H20 ~ &H27	(32 ~ 39)
↓ (アドレスの変化幅は 2 です。)		
8 画のフルーツ色番号	&H833E, &H01 ~ &H0F	(0 ~ 15)
使用するパターン番号	&H833F, &H20 ~ &H27	(32 ~ 39)

パターン番号は 10 進数で 00 から 59 まで使えます。



## ＜早投で競走得の変更＞

### ② 競走得の種類を変える

二つのアドレスの値を合計して、それに 24 を加えた数のパターンから競走得のパターンが選ばれます。

アドレス 1                      &H830A, 1                      (10 点数)

アドレス 2                      &H830B, 3                      (10 点数)

この場合は、 $1 + 3 + 24 = 28$  で、28番から競走得のパターンが選られます。

### ③ 競走得の種類の数を変える

&H830C, 1 ~ 15                      (10 点数)

### ④ 競走得の色を変える

&H8106, 0 ~ 15                      (10 点数)

早投の競走得のアドレスは、&H8106 から 20H ごとです。アドレスの競走得は色番号です。

(10 点数)

＜その他のキャラクターの色を変える＞

④ プレイヤレップ

1P                      &H8008, 0～15                      (10進数)

2P                      &H8028, 0～15                      (10進数)

⑤ 弾の色を変える

                            &H8048, 0～15                      (10進数)

↓

                            &H80E8, 0～15                      (10進数)

20H ごとに6種類の弾が書き込まれています。

⑥ 初期化（変更した変数をすべて元に戻す。）

                            &H8301, 1

または電源を切ります。

## 調整値の変更

### 《調整の出発順序》

アドレス	初期値	変化幅
\$H 8340, 0		0 ~ 3
41, 1		0 ~ 16
42, 2		0 ~ 16
43, 255		0 ~ 16

### 《調整の要素変更》

	アドレス	初期値 (10進数)	変化幅 (10進数)
調整のキャラクタ数	\$H 8350,	5	1 ~ 6
調整のキャラクタ間隔	61,	0	0 ~ 255
並び方変更	62,	0	0 ~ 5
・	63,	0	0 ~ 255
・	64,	55	0 ~ 255
・	65,	66	0 ~ 255
・	66,	3	0 ~ 255
・	67,	2	0 ~ 255
キャラクタ変更	68,	26	0 ~ 59
キャラクタの色変更	69,	4	0 ~ 15

数値の飛び方変更

&H836A, 96

0~255

8B, 240

0~255

以下未定義

シェードゲームは、&H8000以降に書かれています。

POKE文で変更する場合、プログラムにすると便利でしょう。BASICでプログラムの作り方を覚えて応用しましょう。

変更した部分をもとに戻すには、表の初期値を使うか次の命令を実行してください。

POKE &H8301, 1

アドレスは16進数で書いていますが、データは16進数と10進数を用いています。

10 進数      数字だけで書きます。

16 進数      数字の前に &H を付ける。または数字の後ろに H を付けて区別します。

## 2. けいさんボード

計算専用のプログラムです。

BASICでも、計算はできますが、夢数値パーンクであるために、小数点のつく数は計算できません。また、32767以上の数も計算できません。それで、この「けいさんボード」をもちました。

## ＜式の書き方＞

### ★★★ 式で使う記号 ★★★

+	プラス符号 または たし算
-	マイナス符号 または ひき算
*	かけ算
/	わり算
{ }	かっこ

### ★★★ 優先順位 ★★★

- (a) かっこで囲まれた部分 { }
- (b) マイナス符号をつける
- (c) かけ算またはわり算 \* /
- (d) たし算またはひき算 + -

同じ順位の計算記号を二つ以上用いたときは、左側から計算されます。

*note* / コンピュータで計算をする場合、かっこはすべて { } を使います。

{ } のようなかっこは使えません。

### ＜けいさんボードの使い方＞

メニューの2を選び、番号キーを押すと、「けいさんは？」と表示されます。カーソルが点滅していますから、数式を入力して **[CR]** キーを押してください。「こたえは 〇〇〇〇〇」と答えが一行下に表示されます。

せが	けいさん	ボード
けいさんは？	258	× 152
こたえは	49152	
けいさんは？	50-5	× 3+15 / 4
こたえは	28.75	
けいさんは？	画	

一つの計算が終わると、つぎの計算の入力待ちになります。

計算式をまちがえたときは、カーソルをまちがえた所にあわせて正しく打ち直し、**[CR]** キーを押します。

＜計算できる数、計算精度＞

整数部分 8桁に小数点以下 8桁（けた）の計算ができます。8桁を超えると「けいさんできません」と表示がでます。

□□□□□□□□、□□□□□□□□+□□□□□□□□、□□□□□□□□

わり算で割り切れない場合には、小数点以下 9桁目で四捨五入して 8桁目に表示します。

### 3. BASIC

ベーシックでは、楽しいアプリケーションや、実用的な電話帳などのプログラムが作れます。シューティングゲームの背景も描けます。第3章で説明します。

### 4. パターンのへんこう

4のキーを押してください。次の画面があらわれます。これを使って、シューティングゲームのキャラクターを変えてみましょう。

1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 2680, 26

16×16のマス目

カード 15  
しろ

1: カター      2: すすめる      3: もどす  
 4: とうろく      5: しょうぶ      6: いどろーL  
 7: いどろーR      8: いどろーU      9: いどろーD  
 0: ねんでん      ー: ないしょう      H: おいてん

マウス、タッチパッドまたはキーボードを使って、画面左にある 16 × 16 のマス目を白で押めてチャ  
ットを作成します。

機 能	ジョイスティック	キーボード
カーソルを動かす	方向レバー	カーソルキー
点を打つ(白くする)	右のジョイスティック	INS DEL キー
点を消す(黒くする)	左のジョイスティック	HOME CLR キー



## パターンを置く機能、キー番号の説明

1. カラー (.....) キャラクタの色を決めます。色番号と色の名称がキャラクタの下に表示されます。

ここで決めた色を参考にして、ペーノックのプログラムの中で使うとよいでしょう。(カラーは登録できません。参考にするだけです。)

2. すずめる (.....) キャラクタ番号を決めます。キーを押し続けるとキャラクタがつつぎに変わります。

番号は、00～59までの60個分あります。(この番号と、ペーノックで使うパターン番号とは関連があります。P 296参照)

キャラクタの登録やコピーなどしたときは、2のキーを押してもすぐには通みません。2のキーを押して「Y」キーを押すとキャラクタが変わります。

3. もどす (.....) キャラクタ番号を戻します。(番号は記憶していますから、00の後は59,58,57となります。)

4. とうろく (.....) 出東上がったキャラクタを、コンピュータに覚えさせます。  
4のキーを押すと「これでいいですか (Y/N)?」と画面から聞いてきます。登録して良ければ「Y」、まだ変更したいときは、「N」のキーを押します。キーを押すと「これでいいですか (Y/N)?」の表示が消えて登録が終わります。

電源を切るまでは、変更したキャラクタはそのままです。

## Ⅷ. しょうぎょ

画面の消去には5つの方法があります。

3のキーを押す前に、あらかじめ消したい部分へカーソルを移動しておきます。

**[5]** のキーを押すと、“これでいいですか”(Y or N): ALL と表示されます。ALLの文字はカーソルを押すと次のように変わります。キーと意味を覚えましょう。

: ALL 全体を消します。

**[↑]** : UP カーソルのある位置から上方が消えます。

**[←]** : LEFT カーソルのある位置から左側が消えます。

**[→]** : RIGHT カーソルのある位置から右側が消えます。

**[↓]** : DOWN カーソルのある位置から下方が消えます。



コンピュータからの「これでいいですか（漢字していいですか）」の問いかけに “Y” キーを押すと「これでいいですか（Y / N) ?」の表示が消えてキャラクターも消えます。


“N” キーを押したときはキャラクターは消えません。

画し終わったら **[4]** のキーで “とろろく” します。

- |               |                                    |
|---------------|------------------------------------|
| 6. いどろーし…………… | キャラクターが左に移動します。（キャラクターの修正などに使います。） |
| 7. いどろーR…………… | キャラクターが右に移動します。（キャラクターの修正などに使います。） |
| 8. いどろーU…………… | キャラクターが上に移動します。（キャラクターの修正などに使います。） |
| 9. いどろーD…………… | キャラクターが下に移動します。（キャラクターの修正などに使います。） |
| 0. はんでん ……    | キャラクターの白い点の部分が見えなくなります。            |
| - たいしょう ……    | キャラクターを裏返した状態になります。                |
| △ かいてん ……     | キャラクターが 90 度回転します。                 |

180 度回転したときは、たいしょうと違い上下の関係が逆さになります。

④→⑤キーをおすと、画面にある命令が一部変わります。

	キャラクター 番号	
	00	
	<input type="text"/>	
	カラー 15	
	しろ	
1: カラー	2: すずめる	3: もどす
4: とうろく	5: とりかえし	6: コピー
7: くみあわせ	-: リスト	8: ごうせい

(スペースキー) 5 とりかえし

画ページの5のキーで、うっかりキャラクターを消してしまったときは、これでキャラクターを取り返すことができます。しかし、4のキーで登録した場合は取り返せません。

(スペースキー) 6 コピー

あるキャラクターを別のキャラクター番号の所へコピーします。例として、キャラクター 20 を 41 にコピーします。

- ① --- まず、コピーしたい 20 番のキャラクターを画面にだしておきます。
- ② ----- 6のキーを押すと  
「コピー: FROM 20 TO 20」

「これでいいですか。(Y/N)?」

と画面から聞いてきます。

- ② ー——— そこで、2のキーをおして TO 30 の番号を 4) になるまで押します。  
(4) には何も描かれていません。
- ③ ー——— 番号が 4) になったら "Y" キーを押します。
- ④ ー——— 30 のキャラクターがコピーされてきます。
- ⑤ ー——— 4のキーで登録してください。登録を忘れると消えてしまいます。
- ⑥ ー——— これでコピーは終わりました。

(スペースキー) - リスト

「ハターアのへんこう」画面でつくったキャラクターのデータを与えてくれます。データは次のように、16進数であらわされます。

# 160	→	01	01	02	02	05	05	09	09
# 161	→	05	05	15	20	66	E8	EA	D7
# 162	→	00	00	80	60	40	40	20	20
# 163	→	C0	A0	50	A8	2C	AE	AE	D6

このデータは、ページの PATTERN 文に入れて使うことができます。

例

PATTERN B # 160 , 

↑ この部分に上記のデータを入力

note / 16進数や、PATTERN 文については、第3章 ページの PATTERN 文、SPRITE 文の項をご覧ください。

## 《スペースキー》⑧ ごうせい

4個までのスプライトを合成したときの状態を確かめられます。画面画面ごとにも色分けして、それを合わせるとカラフルなキャラクターが作れます。

① ーーーー 作りたいキャラクターの下絵をグラフ用紙などに16×16のマス目を書いて、それを4個以内に分けます。

② ーーーー 分けた部分をも、別々のキャラクター番号に登録します。

たとえば、40, 41, 42, 43に登録しておきます。

登録した40番のキャラクターを出しておきます。

③ ーーーー 1のキーでカラーをきめます。

④ ーーーー ごうせいのキーを押すと、つぎの表示がでます

「40 T O 01」

「これでいいですか (Y/N)?」

⑤ ーーーー 1のキーを押します (仮りにスプライト番号を1にきめる。)

⑥ ーーーー “Y”キーを押すと、色番号の下にキャラクター番号とキャラクターが現われます。  
これで40番のキャラクターがスプライト1となりました。

⑦ ーーーー つぎに、“すすめる”で41番のキャラクターを出して、⑥から同じ手順でスプライト番号を1, 2, 3, 4とします。

(同じスプライト番号を使うと、前のキャラクターが消えてしまいます。)

⑧ ーーーー 必要な回数を繰り返してください。

ここで使ったキャラクターは、参考にするだけではかには使えません。

作成したキ+ラタタを直すには、空白を“ごうせい”と同じ手順で直すが、BREAK キーでオープニングに戻します。

### 〔スペースキー〕? くみあわせ

2つのキ+ラタタをくみあわせたい形を見たいときに使います。ここでくみあわせて作った形も、実際にするだけで他には使えません。

例として、キ+ラタタ 28 と 30 をくみあわせてみましょう。

① …… まず、くみあわせたいキ+ラタタ2つのうち片方(28)を画面に出しておきます。

② …… ? のキーを押すと、

「くみあわせ: FROM 28 TO 28」

「これでいいですか (Y/N) ?」

と両面から聞いてきます。

③ …… そこで、2のキーを押して TO 28 の番号を、30になるまで押します。

④ …… 番号が 31 になったら “Y” キーを押します。

⑤ …… 28 と 30 のキ+ラタタがくみあわされます。

## 第 3 章 BASIC

人間の言葉には、日本語、英語、フランス語など、たくさんの言葉があります。

コンピュータも、たくさんの言葉を持っています。ペーラックはその中の一つです。ペーラックを使えば、コンピュータにいろいろな仕事をさせることができます。

ペーラックは、英語ですがけっこうむずかしいものではありません。毎日少しずつ使っていると自然に覚えてしまいます。では、やさしいところからはじめましょう。

### ダイレクト・モード (直接命令)

命令のしかたには、直接命令 (ダイレクト・モード) と間接命令 (一般に、行番号がついており、「プログラム」とよばれています。) の二種類があります。

ここでは簡単な方、つまり、直接命令について説明しましょう。(プログラムは、のちほど扱います。)

### PRINT (プリント)

ダイレクト・モードで、コンピュータに PRINT という命令を実行させましょう。

PRINT とは、コンピュータが仕事をした結果を画面に知らせる命令です。

文字や記号を書く

BASIC TEST と書いてみましょうか。

次の手順で、コンピュータを操作して下さい。



**PRIME/CLE** キーをおします。

PRINT " BASIC **SPACE** TEST " と入力してください。

このとき、ダブルクォーテーション " " を忘れずに付けてください。(シフトキーを押したまま

**3** のキーを押せば " " が出ます。

**CR** キーを押し、実行させます。

そうすると、次のような画面になります。

```
PRINT " BASIC  TEST "
```

```
BASIC  TEST
```

```
Ready
```



← BASIC TEST を画面に出力と命令しました。

← 実行しました。

← 次の命令を待っています。

**note 1** : コンピュータに文字や記号を書かせる場合は、必ず、" " (ダブルクォーテーション) をつけてください。

**note 2** : PRINT と打ちこむ代わりに、? を使っても、同じ働きをします。

## **HOME/CLR** ? \* BASIC TEST \* **CR**

と入力しても、結果は同じです。

*note3 /* : Syntax Error (シンタックス・エラー) について

BASIC 命令を正しく書くためには、シンタックス (構文) という約束を守らなければなりません。BASIC 命令を書き込んで実行したとき、もしも

? Syntax Error

Ready

画

と画面に出たら、キーを打ちまちがえているということですから、画面の文字を見直してください。つづきを少し間違えても、コンマの代わりにピリオドを使ってしまった場合にも、エラーになります。正しく入力するよう心がけてください。

### ＜簡単な計算＞

コンピュータに四則計算をさせましょう。

4 + 3 の計算をしましょう。次のような手順をとります。

**HOME/CLR** キーを押します。

**PRINT** 4 + 3 と入力します。

(または、? 4 + 3 のように、? を使ってもかまいません。)

**CR** キーを押して実行させます。

そうすると、画面は次のようになります。

```
PRINT 4+3
```

```
?
```

```
Ready
```



← 4+3 の計算結果を出せと命令しました。

← 実行して、答えを出しました。

← 次の入力を待っています。

*note1 /* : 計算のときは、ダブルクォーテーション " " は必要ありません。

*note2 /* : コンピュータで計算する場合、計算に使う記号 (演算子) は、一部普通の計算と違います。

たし算 + (プラス)

ひき算 - (マイナス)

かけ算 \* (アスタリスク)

わり算 / (スラッシュ)

*note3 /* : わり切れないわり算について

たとえば、10/3 の計算をすると、答えは 3 と出ます。小数点以下は切り捨てられるのです。

MOD (モド) を使うと、わり算の余りを出すことができます。

? 10 MOD 3 と入力すると、1 と余りが表示されます。

note4 / 計算は、-32768 から 32767 の範囲内でのみ可能です。その範囲をこえる場合は、(「けいさんボード」)を使ってください。

note5 / 四則演算には優先順位があります。それについては、けいさんボードの項をご覧ください。

#### ＜PRINT 命令の応用＞

これまでに、PRINT の二つの使い方を説明しました。

それは文字の表示と、計算の二つでした。それらを組み合わせると、こんなことができます。

```
PRINT "4+3=" ; 4+3
```

```
4+3=7
```

← "4+3=" を文字として表示して、4+3 の計算結果を表示せよと命令しました。

← 4+3= が表示され、答え(7)が出ました。

note1 / 文字として扱う 4+3 には必ず " " (ダブルクォーテーション) をつけてください。文字として扱う部分と、計算式の部分を区別するために、; (セミコロン) を忘れずにつけて下さい。

note2 / セミコロンのかかわりに、, (カンマ) を使うと、両端の端から8格離れた場所に答が表示されます。

(例) PRINT "2+3=" , 2+3

2+3=

5

8 格

## プログラム・モード その1

### あなたの名前をコンピュータが書きます

あなたの名前は何ですか。ハナコさん？ それとも アキコさん？ これから、コンピュータに命令して、あなたの名前を書かせてみましょう。

まずは、ダイレクト・モードを使いましょうか。キキほど、BASIC TEST と呼ぶのと同じことです。

仮に、アキコ という名前とします。

```
HOME/CLR
```

```
PRINT "アキコ" CR
```

**CR** キーを押すと、画面には アキコ と表示されますね。

ダイレクト・モードでは、**CR** キーを押して命令を実行してしまうと、それきりおしまいです。もう一度名前を書かせたいと思ったら、または始めからコンピュータに打ちこまなければならぬのです。何度も同じことをやりたくないとき、これでは不便です。どうしましょうか。

次のように入力してください。(各行の終わりに、かならず **CR** キーを押してください)

```
NEW
```

```
10 PRINT "アキコ"
```

```
20 END
```

これから、アキコと書かせるためのプログラムです。これを実行させるには、**RUN** と入力し、

**CR** キーを押します。そうすると、

アキコ

Ready



と画面に出ます。命令が実行されたのです。プログラムは、パソコン本体のスイッチを切らない限り、何度でも、RUN と入力すれば同じように実行されます。実際に、何度も RUN と入力してみてください。

アキコ

Ready

RUN ————— ( **[CR]** キーを押す )

アキコ

Ready

RUN ————— ( **[CR]** キーを押す )

アキコ

Ready

RUN ————— ( **[CR]** キーを押す )

;

と、RUN するたびにコンピュータは何度でも同じように名前を書いてくれます。「1度だけ」と「何度でも」——これがダイレクト・モードとプログラムモードとの大きな違いなので、もうひとつ注目してほしいことがあります。各行の左側につけた番号です。これは行番号と呼ばれ、

プログラムはその番号の順序で実行されます。行番号に10000は、後でもう少し詳しくみれます。

さて、名前もひとつだけではつまらないですね。今度は、たくさん名前を書かせてみましょう。次のプログラムを打ちこんでください。

(プログラムを打ちこなときは、各行の終わりに必ず **CR** キーを押す習慣をつけましょう。)

```
NEW  
10 PRINT "アキコ"  
20 GOTO 10  
30 END
```

プログラムが完了したら RUN してください。

```
アキコ  
アキコ  
アキコ  
.
```

と、続にずっと繰り返すね。

**STOP** / コンピュータは、いつまでも「アキコ」を書いていきます。止めるときは **BREAK** キーを使ってください。

**GOTO** は、「指定された行番号(ここでは10です)へとべ」という命令です。コンピュータは忠実にそれを実行し、アキコという名前を、続に書き続けたわけです。

では、決められた回数だけ、名前を書かせるためにはどうしたらよいでしょうか。

次のプログラムを打ちこんでください。

```
NEW
10 FOR N=1 TO 5
20 PRINT 'アキコ'
30 NEXT N
40 END
```

打ちこんだら RUN してください。画面には、

```
アキコ
アキコ
アキコ
アキコ
アキコ
```

と、縦に五つアキコが並びます。これは名前を五つだけ書かせるプログラムなのです。ここで、FOR ~ NEXT という命令が使われていますが、この命令は一定回数、繰り返し仕事をさせる命令です。

さらに、ちょっと趣んであなたの名前で画面をいっぱいにしてみましょう。次のプログラムを打ちこんでください。



```

NEW
10 PRINT 'アキコ' ;
20 GOTO 10
30 END

```

さあ、RUN してください。どうなりましたか？

*note 1* PRINT 文で ; を使うと、文字をすぐ横に並べて表示します。

これまでに、あなたの名前をひとつ横かせたり、たぐさん横かせたり、横に並べたり、縦に並べたりしましたが、それらのプログラムの中には、いろいろな命令がでてきましたね。ここで、もう一度整理しておきましょう。

NEW ( ニュー ) …… プログラムを打ちこむ前に必ずこれを入力してください。前にあったプログラムを消します。

PRINT ( プリント ) …… 画面に、プログラムの実行結果を表示します。

*note 1* 文を区別るときは、または ; をつかいます。

*note 2* PRINT **CR** とすると、改行します。

END ( エンド ) …… プログラムが終わりであることを示します。

RUN ( ラン ) …… 打ちこんだプログラムを実行させます。

[例] RUN …… プログラムを先頭から実行する。

RUN 100 …… 行番号 100 から実行する。

GOTO (行番号) ; 指定された行番号に飛びます。GOTO で指定すれば、どこにでもとどくことができます。

(例)                      行番号

	→ 10	前へもどります。
50 GOTO	→ 50	同じところにいます。[グラフィック画面でよく使います。]無限ループをつくらといひます。
	→ 100	先にとびます。

FOR ~ NEXT (変数 ~ 終値) ~~~~~ 一定回数、仕事を繰り返します。

FOR ~ NEXT ~ STEP ~ (変数 ~ 終値 ~ ステップ) ~~~~~ 一般に、

FOR	●	=	□	TO	□	STEP	□
	↓		↓		↓		↓
	変数		初期値		最終値		変化幅

~~~~~                      ← 繰り返させたい仕事のプログラム

NEXT                      ●

の形であらわれ、ある仕事を、変数が初期値から最終値まで変化する間だけ繰り返します。STEP は変化の幅を指定します。

たとえば、

```
10 FOR N=0 TO 10 STEP 2
20 PRINT N:
30 NEXT N
```

とすると

0 2 4 6 8 10

のように、(変化幅が2だから)2つおきに数字が並びます。

また、

```
10 FOR N=10 TO 0 STEP -2
20 PRINT N:
30 NEXT N
```

とすると、

10 8 6 4 2 0

のように、(変化幅が-2だから)2ずつ減って、数字が並びます。

STEP 1は省略可能です。さきほどの例(アキコと出会えるプログラム)では、STEP 1が省略されていたわけです。

```
10 FOR N=1 TO 5 (STEP 1)
20 PRINT 'アキコ'
30 NEXT N
```

は、変数Nが1から5まで変化する間(つまり、5回)、PRINT 'アキコ'という仕事を繰り返すのです。

FOR～NEXT～STEP文は、二重、三重に重ねることができます。  
(8回まで可能です。)

```
10 FOR J=1 TO 3  
  20 FOR I=1 TO 3  
    30 PRINT 'アキコ';  
    40 NEXT I  
    50 PRINT  
  60 NEXT J
```

このプログラムを実行すると

アキコアキコアキコ

アキコアキコアキコ

Ready



となります。これを「入れ子」といいます。

その他に次のような命令文もあります。

CLS …… 画面上の表示を消す命令です。表示されているものを用いても、プログラムはメモリの中に残っています。その点が NEW と違うのです。

*note* / プログラムの先頭に入れておけば、画面を消してから実行します。

STOP …… プログラムの実行を途中で止めたいときに使います。

CONT …… STOP や BREAK キーで止めた後、再びプログラムを続けたいときに使います。

STOPとCONTを使いた

サンプルプログラムです。

```
10 REM - STOP と CONT :  
20 FOR N=1 TO 100  
30 PRINT N;  
40 IF N<>50 THEN GO  
50 PRINT:PRINT  
60 PRINT"CONT と 9999"  
70 STOP  
80 NEXT N  
90 END
```

RUN

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22  
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  
43 44 45 46 47 48 49 50
```

CONT と 9999

Break in 70

CONT

```
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69  
70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
90 91 92 93 94 95 96 97 98 99 100
```

Ready

## プログラムは保存できます

さきほど、ダイレクトモードとプログラムの違いを述べました。プログラムは、同じことを何度でも繰り返し実行できて便利だといいました。けれども、それは、「パソコン本体のスイッチを切らない限り」という条件付きでしたね。せっかく作ったプログラムがスイッチを切ったとたんに消えてしまうなんて、とがっかりしているあなたが目にうかびます。でも、大丈夫。プログラムは保存ができるのです。

### SAVE(セーブ)

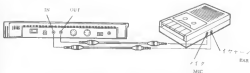
セーブとはカセットテープにプログラムを録音することです。

プログラムはカセットテープに保存しておくことができます。次の手順でやってください。

- セガデータレコーダ SR-1000 を使う場合



■ 他社のカセットレコーダを使う場合



カセットレコーダ用のケーブルは図のような、内側がニブナゲになったものを使います。  
 別売製品のケーブル以外を、お使いになるときは、既読の入っていないものをお使いください。

SAVE <sup>①</sup>XXXX<sup>②</sup>と入力します。(XXXX には、プログラムの名前を入れます。)

**[CR]** キーをおします。

Saving Start の表示がでたら、オーディオカセットの録音ボタンを押します。

約8秒間にピーと音がして SAVE が始まります。

Saving End と、記録が終わったことを知らせる表示がでたら、カセットを止めます。

テープによっては、書き始めの部分に録音できない所があります。少し巻いてからテープしください。

**note 1/** コイルネーム(プログラムの名前)に代えて、プログラムの内容がわかる名前を16字以内でつけてください。

**note 2/** カセットデッキはお手持ちのものも使えます。カセットデッキの電源スイッチが壊れていましたらカウンターの数字をカセットテープのラベルに記入しておいて下さい。お手持ちのカセットデッキを使う場合、音のレベルにより、カセットテープに書き込みや、読み取りが出来ない場合があります。これは、コンピュータの故障ではなく、カセットデッキの性能による場合があります。

## VERIFY(バリファイ)

SAVE(プログラムの保存)が正確に行われたかをチェックするには、次のようにします。

- ・SAVE を始めた位置までテープを巻き戻します。
- ・VERIFY と入力して、**[CR]** キーを押します。
- ・カセットデッキのプレイボタンを押してください。

正しくSAVE されていれば、Verify end と表示されます。

end が出ない場合は、RESET キーで始めて、初めからSAVE したおししてください。

## LOAD(ロード)

カセットテープにSAVE されているプログラムをとり出し、コンピュータに移すことをロードするといいます。



LOADは、次のようにします。

LOAD "XXXX" (XXXXはプログラム名)と入力して **[CR]** キーを押します。

LOAD **[CR]** のようにプログラム名は省略できます。

Loading Start と表示が出たら、カセットデッキのプレイボタンを押してください。

Found "XXXX"

Loading End と表示が出たら、カセットを止めてください。

*note* / ープリンタを持っている方へー

プリンタ用紙に印字して、目に見える状態で残すこともできます。

## LIST (ユルリスト)

プリンタにプログラムリストを書かせるのは、LIST 命令です。使い方は次のとおりです。

LIST : プログラム全図を書きます。

LIST 行番号 : 指定した行番号を書きます。

LIST 行番号 - 行番号 : 指定した行番号から行番号までを書きます。

LIST -行番号 : 先頭から指定した行番号までのリストを書きます。

LIST 行番号- : 指定した行番号以降のリストを書きます。

文字や数値の値をプリンタで印字する命令は LPRINT です。使い方は PRINT と同じです。

LPRINT "セガ"

と入力すれば、セガとプリントアウトされます。

## 知っていると便利です

### ★★★ プログラム内容の確認 ★★★

何かプログラムをつくられたとします。それが正しく組まれているか、修めたいときは LIST と入力して [CR] キーを押して下さい。プログラムが表示されます。これを「リストを出す」といいます。アスキーと書かせるプログラムを例に見ますが、プログラムを END にした後、あるいは、RUN した後、LIST と入力しますと、

```
10 PRINT "アキコ"  
20 END
```

と、プログラムの全内容が表示されます。表示された内容を見て、訂正したいところは訂正し、つけ加えたいことはつけ加えます。

*note? /*

LIST の使い方は、次の方法があります

|      |          |                         |
|------|----------|-------------------------|
| LIST |          | プログラムの内容を全部表示します。       |
| LIST | 行番号      | 1 行のみ表示します              |
| LIST | 行番号- 行番号 | 行番号から行番号までを表示します。       |
| LIST | 行番号-     | 行番号から後ろのプログラムの内容を表示します。 |
| LIST | - 行番号    | プログラムの先頭から行番号までを表示します。  |

note2 /

LIST したとき、画面いっぱいになるほどプログラムの行数が多いと、リストはせり上がっていきます。

それをとめてリストを見たいときは **[SPACE]** キーを押してください。もう一度 **[SPACE]** キーを押すと、LIST は再開します。

### \*\*\* プログラムの訂正 その1 \*\*\*

LIST でプログラム内容を確かめるとき、何か間違いを発見したら、次のようにして訂正してください。

カーソルを、訂正したい文字(あるいは数字、記号)の上に移動させます。そこに、正しい文字(あるいは数字、記号)を打ち込んでください。

正しく書き替えたら **[CR]** キーを押してください。

note1 / インサート・デリートの項目も参照してください。

note2 / せりあがっていく長いプログラムリストの中に間違いをみつけたときは、**[BREAK]** キーを押して下さい。その時点で LIST の動きがとまり、カーソルが点滅します。カーソルを動かして、訂正してください。

### \*\*\* プログラムの訂正 その2 \*\*\*

LIST で出したプログラムを見て、一行を消したいときは、行番号をけ入力し、**[CR]** キーを押します。

<例> 20 行を消したいとき

20 **[CR]**

### \*\*\* 行番号のこと \*\*\*

プログラムの各行に番号をつけるということは、前に述べました。行番号は、たいてい、10 20 30 40 …… というように、10きざみです。どうしてでしょうか。

行番号を、ある程度間隔をあけてつけるのは、あとで新しく行をつけ加える場合、その方が便利だからです。

たとえば

10    x x x x

20    a a a a

この2行の間に、□□□□という一行を加えたい場合には10    □□□□ というように、10 と 20 の間にある数字を、行番号にします。もし、假に1    2    3    4 …… というように行番号がついていたなら、新しい一行をつけ加えることができないのです。

**note /** 行番号の範囲は1から 65535 までです。

\*\*\* 新しい一行をつけ加える法 \*\*\*

```
NEW
10 PRINT "アキコ"
20 END
```

のプログラムの 10 と 20 のあいだに GOTO 10 という一行をつけ加えたいときは、10 GOTO 10 と入力して **[CR]** キーを押して下さい。(図 ①)

行番号

念のため、LIST してみましょうか。図 ② のようになりますね。

```
NEW
10 PRINT "アキコ"
20 END
10 GOTO 10
```

Ready



図 ①

```
LIST
10 PRINT "アキコ"
10 GOTO 10
20 END
```

Ready



図 ②

番号順に入力しなくても、ちゃんと小さい番号から並びます。

\*\*\* 自動装置 AUTO (オート) \*\*\*

行番号を打ち込むのがめんどろな方、**AUTO**を使いましょう。行番号を自動的に増してくれま  
**AUTO**と入力して **[CR]** キーを押してください。自動的に 10 と表示されましたね。行番号 10 のとこ  
ろに何かプログラムを入力し **[CR]** キーを押すと、今度は 20 と表示されます。このように、10 20  
30 40 …… と、10 きざみに自動的に行番号が表示されています。

また、次のようにも命令することができます。

**AUTO 100 [CR]** …… 行番号が、100 番から 10 きざみで発生

**AUTO 10,20 [CR]** …… 行番号が、10 番から 20 きざみで発生

(10 からの発生、きざみ 10)

**AUTO** の機能を止めるには、**[BREAK]** キーを押します。

\*\*\* 見出しをつけよう REM (レム) \*\*\*

プログラムを作成するとき、注釈文 (コメント) を入れておくと、後でプログラムリストを見たとき  
便利です。ちょうど、新聞などの見出しのような役割を果たしてくれます。

```
10 REM *** タイマン ***
```

```
20 PRINT 2+3
```

```
↑
```

上記のように、**REM** という命令を使います。**REM** 文の行は実行されません。

注釈文は、プログラムの内容、製作者の名前、制作した年月など、あなたが後で見易いように工夫  
してください。 note / REM はリマークとも呼びます。

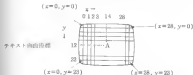
# \*\*\* 画面上のレイアウト 17

CURSOR, SPC, TAB, コンマとセミコロン \*\*\*

## ＜CURSOR(カーソル)＞

ふつう、プログラムをRUNすると、画面の左側に表示されますね。でも、表示の位置を設定することもできるのです。CURSOR という命令です。

テキスト画面(プログラムを書く画面)は次のように 28×24 の 696 のマス目で構成されています。



x(横)とy(縦)の座標指定によって、好きなように表示する場所を指定できます。

CURSOR x軸, y軸

と入力して下さい。たとえば、上の図の位置にAを表示するには、

CURSOR 14,12:PRINT 'A'

と入力して **[CR]** キーを押します。

*note* / CURSOR 文で座標を指定すると、表示したい文字の先頭位置が決まります。

#### <SPC(スペース)とTAB(タブレーション)>

文字と文字の間にスペースを設けたい場合は、SPCで指定します。

```
10 PRINT "ABC"; SPC(10); "XYZ"
RUN
ABC          XYZ
           10文字分の空白(スペース)
```

TABは、画面の端から何文字目に出すかを指定します。

```
10 PRINT TAB(5); "ABC"
RUN
      ABC
    5文字分のスペース
```

*note1* / SPCもTABも、PRINT文で使います。

*note2* / SPCで指定した範囲に文字があると消されてしまいますが、TABの場合は、文字があっても消されません。



### <セミコロンとコンマ>

セミコロンは、PRINT 文の区切りに使います。セミコロンで区切ると、実行したとき、横並した形で横にならびます。コンマを使うと、区隔を置いた形で横にならびます。(下記参照)

```
PRINT "A"; "B"; "C"; "D"; "E"; "F"
```

```
RUN
```

```
ABCDEF
```

```
PRINT "A", "B", "C", "D", "E", "F"
```

```
RUN
```

|   |     |   |  |   |  |   |
|---|-----|---|--|---|--|---|
| A | 8文字 | B |  | C |  | D |
| E |     | F |  |   |  |   |

ここで、HOMIE BASICで使われる記号について、説明しておきましょう。わかりやすくするため、表にしてみました。

| 記号名                                 |  |                                                    |
|-------------------------------------|--|----------------------------------------------------|
| セミコロノ ( <code>;</code> )            |  | PRINT文の区切りに使います。                                   |
| コンマ ( <code>,</code> )              |  | PRINT文、INPUT文、DATA文で、区切りとして使います。                   |
| コロノ ( <code>:</code> )              |  | 文の区切り記号。1つの行番号の中に2つ以上の文を書くとき使います。〔マルチ・ステートメント〕     |
| ダブルクォーテーション<br>( <code>" "</code> ) |  | この記号でかこまれた文字は、文字のかたまり〔文字列〕として扱われます。                |
| ドルマーク ( <code>\$</code> )           |  | 文字列変数につけて数値変数と区別します (PLAY文のフラット記号)                 |
| 疑問符 ( <code>?</code> )              |  | PRINT文の代用です。                                       |
| 角記号またはハイフン<br>( <code>&lt;</code> ) |  | 角記号、引き算に使います。また、LIST文、DELETE文で行の指定に使います。           |
| スペース<br>(スペースキーを押す)                 |  | スペースを避けるときに使います。<br>(スペースも文字のひとつとして扱われます。)         |
| アンド ( <code>&amp;</code> )          |  | Hとあわせて、&HPのように使用して、16進数をあらわします。                    |
| シャープ ( <code>#</code> )             |  | PLAY文で使います。(PLAY文のノープ記号)<br>SPRITE、PATTERN文でも使います。 |

記号を〔特にセミコロノとコロノ〕使い分けて、見易い表示を心がけましょう。

\*\*\* その他の知っている便利なこと \*\*\*

＜残っているメモリは？ FRE(フリー)＞

プログラムを入力すると、メモリがどんどん減っていきます。あととれくらい残っているか知りた  
いときは、FRE関数で調べます。

例

```
PRINT FRE  [CR]
6538
```

あと 6538 BYTE(バイト)のプログラムを入力できることを示しています。  
(バイトは単位のひとつです。)

＜カーソルの移動範囲の設定 CONSOLE(コンソール)＞

たとえば、

```
CONSOLE 5, 15
```

と入力すると、5行から15行まで、15段の間をカーソルが移動します

また、CONSOLEは、ブロック音をONにしたり、OFFにすることや、英文字の大小切り替えの指  
定もできます。

一般に次のような形をとります。

```
CONSOLE V, L, C, S
```

V: スタートルの上限 (0~32)

L: スタートルの長さ (2~24)

C: ブロック音の有無 <sup>0: 無し</sup>  
                          1: 有り

9: 英数のみ 0: すべて大文字  
1: すべて小文字

— 解除する場合 —

CONSOLに 9, 24, 1, 9 でもとの状態に戻ります。RESET キーを押しても、もっにもどります。

## プログラム・モード その2

### もっといろいろ、計算できます

\*\*\* LET, 変数, INPUT \*\*\*

ダイレクト・モードのところで、四則計算を行いました。たね、 $4+3$  の計算は、**HOME/CLR**  
**PRINT 4+3** **CR** でした。

この計算は、プログラム・モードでは

```
10 LET A=4
20 LET B=3
30 LET C=A+B
40 PRINT C
50 END
```

のように入力します。RUN して実行させると、7 と答えが出ます。

LET という新しい命令ができましたが、これは「代入する(変数に数値または文字を入れる)」命令です。さきのプログラムの、 $A=4$  は  $A$  という変数に 4 を代入する、 $B=3$  は  $B$  という変数に 3 を代入する、 $C=A+B$  は  $C$  という変数に  $A+B$  を代入する、という意味なのです。

この場合、注意しなければならないことは、必ず、変数を左辺に置くということです。 $4=A$  ではエラーになります。気をつけましょう。

LET は省略することができますから、さきのプログラムは

```
10 A = 4                40 PRINT C
20 B = 3                50 END
30 C = A + B
```

とすることもできます。

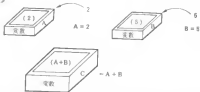
#### 4 INPUT (インプット)

さて、 $4+3$  の計算だけでなく、 $1+2$  でも  $4+8$  でも、二つの数を加える計算なら何でもできる便利なプログラムにしましょうか。次のように入力してください。

```
10 INPUT A
20 INPUT B
30 C = A + B
40 PRINT C
50 END
```

INPUT は、数値、あるいは文字を入力してくださいという命令です。RUN で実行させると、コンピュータは ? という表示を出して、何か数を入力してくださいと要求してきます。あなたが何か (たとえば 3) を入力し、**[CR]** キーを押すと、また ? が出て、今度は B に数を入力してくださいと要求します。あなたが何か (たとえば 8) 入力し、**[CR]** キーを押すと、コンピュータは  $A+B$  の計算をして答え (この場合 7) を出してくれます。

< 変数 >



INPUT と ? は割り切られていると考えてください。? が表示されたら、あなたは何かを入力しなくてはなりません。

ところで、INPUT A の A は、さきほどふれた「変数」なのですが、変数にはもう一種類、「文字変数」というものがあります。それは一般に A\$ のように、アルファベットに \$ (ドルマーク) がついた形であらわされます。INPUT A\$ と入力して **[CR]** キーを押すと、やはり ? マークが表示されます。

文字変数を使って、名前をたずねるプログラムをつくりましょう。

NEW

10 INPUT " 名変え様 " ; A\$

```
20 PRINT A$;"さんですね"
30 END
```

これを実行させると

なまえは?

ときいてきます。そこで、あなたが自分の名前をタイプする

と。(ひとしとしましようか、)

ひとしさんですね

Ready



と画面に出ます。

*note1 /* : INPUT 文は、プログラムの実行中にキーボードから数値や文字を変数に入れるときに使いますが、その際、INPUT 文に続く変数の形に気をつけましょう。

INPUT X

数値しか入れられません。(Xは数値変数)

INPUT A\$

文字や記号を入れます。数値を入れたときは文字として扱われます。  
(A\$は文字変数)

*note2 /* : 2つ以上の文字変数を用いることができます。

10 INPUT A\$ ← 数値を入れてください。(123を入れてみる。)

20 INPUT B\$ ← 数値を入れてください。(456を入れてみる。)

30 C\$ = A\$ + B\$ ← たし算ではなく数字をつなげます。



40 PRINT C\$ ← 123456 とつながります。

50 GOTO 10 ← はじめに戻ります。 文字も入れてみましょう。

数字と文字をつなげることもできます。

計算の話にもどります。GOTO を計算プログラムに利用すると、とても便利です。やってみましょう。たし算のプログラムに GOTO を使った一行をつけ加えて、次のように入力して下さい。

```
10 INPUT A
20 INPUT B
30 C=A+B
40 PRINT C
50 GOTO 10
60 END
```

*note /* : INPUT 文は、キーから入力があるまで、? を表示して待っています。

このようにすると、そのつど RUN しなくても、A と B に数値を与えてあげれば、コンピュータは次々に計算して答えを出してくれます。

## プログラム・モード その3

今までは、コンピュータの仕事をいっでも、ほんの簡単なことでした。でも、IFを使わなくてもできるようなことでした。でも、こんなこともできるんですよ。

### 合格と不合格に振りわけます

★★★ IF ～ THEN ★★★

IF ～ THEN は条件判断をします。ある条件に符って、次にすすむべき方向を指定するのです。具体的に、合格、不合格に振りわけるプログラムをつくってみましょう。

```
10 INPUT A
20 IF A < 65 THEN 100
30 PRINT "ごうかく"
40 GOTO 10
100 PRINT "ふごうかく"
110 GOTO 10
```

実行するとコンピュータは、まず、「とくてん？」と聞いてきますから、あなたは得点を入力して下さい。入力した得点が 65 点以上であれば「ごうかく」、65 点に満たないと「ふごうかく」と表示されます。

このような、ある時点で分岐するプログラムは、フローチャート（流れ図）に書くと、よくわかります。



IF ～ THEN は、次のように使うこともできます。

|                     |              |
|---------------------|--------------|
| IF ～ THEN GOSUB 行番号 | (指定行番号にとびます) |
| IF ～ THEN PRINT “ ” | (画面に表示します)   |
| IF ～ THEN END       | (プログラム終了)    |
| IF ～ THEN STOP      | (プログラム実行中止)  |
| IF ～ THEN BEEP      | (音を出します)     |

## ゴテュウモンは?

\*\*\* どこにとが ON → GOTO \*\*\*

IF → THEN と似ているものに、ON → GOTO があります。

IF → THEN は、

IF 未 許 THEN 実行すべき仕事

の形をとり、もし   なら   せよという命令です。

それに對して、ON → GOTO は、

ON 3 5 GOTO 行番1 , 行番2 , 行番3 . . .

の形をとります。変数が1であれば、GOTO の後に並んでいる1番目の行番へ、変数が2であれば、2番目の行番へとんで仕事を実行せよという命令なのです。IF → THEN か、ある条件によって行先が決まるとすれば、ON → GOTO は、変数の値によって行先が決まるのたといえます。

では、実際に ON → GOTO を使ってプログラムをつくりましょう。次のプログラムは、表示されたメニューの中から何か品物を選ぶと、その値段がでてくるというプログラムです。

```
5 CLS
10 PRINT "メニュー"
20 PRINT "1・・・コーヒー"
30 PRINT "2・・・ジュース"
40 PRINT "3・・・ケーキ"
50 PRINT "INPUT 1～3"
```

```

60 INPUT "ゴチャウモンは?" : A
70 ON A GOTO 100, 200, 300
80 GOTO 10
100 PRINT:PRINT "コーヒー・・・¥250"
110 PRINT:GOTO 10
200 PRINT:PRINT "ジュース・・・¥300"
210 PRINT:GOTO 10
300 PRINT:PRINT "ケーキ・・・¥350"
310 PRINT:GOTO 10

```

※※※ 100, 110, 200, 210, 300, 310 行のはじめの PRINT は、プログラムに  
一行分の空白をつくら、見やすくするためのものです

このプログラムを実行すると、画面は次のようになります。

```

メニュー
1・・・コーヒー
2・・・ジュース
3・・・ケーキ
ゴチャウモンは?

```

あなたが1を入力すると、100行が実行され、

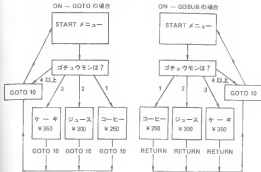
コーヒー・・・ ¥ 250

と表示されます。2を入力すると200行、3を入力すると300行が実行されます。

ON ~ GOTO を使った上記のプログラムは、ON ~ GOSUB を使って次のように書き換えることができます。ON ~ GOSUB は、RETURN と組みで使います。

```
10 PRINT "メニュー"
20 PRINT " 1・・・コーヒー"
30 PRINT " 2・・・ジュース"
40 PRINT " 3・・・ケーキ"
50 PRINT " 1～3までのかずをいれてください"
60 INPUT "ゴチュウモンは？" : A
70 ON A GOSUB 100,200,300
80 GOTO 10
100 PRINT:PRINT "コーヒー・・・ ¥ 250"
110 RETURN
200 PRINT:PRINT "ジュース・・・ ¥ 300"
210 RETURN
300 PRINT:PRINT "ケーキ・・・ ¥ 350"
310 RETURN
```

このプログラムを図にすると次のようになります。



ON ~ GOTO の場合1、プログラムのはじめ (行番号 10) にもどすには、GOTO を使いますが、ON ~ GOSUB の場合は、RETURN で GOSUB の次の行にもとります。

*note /* : GOSUB で指定されるとぶ先を「サブルーチン」といいます。



### デジタル時計に早があり ?

コンピュータの内部には、本製品版の正確なデューノ・デジタル時計の機能があります。次のプログラムを実行したら、

```

10 TIME $ = " 00:15:00 " ← 現在の時刻 (" 時:分:秒 ")
20 CURSOR 15,15          ← 表示する場所を設定します。
30 PRINT TIME $          ← 時刻を表示しなさいと命令しました。
40 GOTO 20                ← 時を刻み続けます。
  
```

コンピュータは時計に化身です。

*note /* : 解除するには、RESET を押します。



note 2 / : ディレクトリ・カードで

PRINT TIME \$

と入力して **[C]** キーを押すと、電源を入れてから、または RESET してからの時間が表示されます。

### たくさんのデータも... READ, DATA

たくさんのデータの処理も、READ, DATA 文を使えば簡単です。

この二つの命令はいつでも組みで使います。

DATA 文でデータをあらかじめ入力しておき、READ 文でそれを読みます。DATA 文をどこに置いても、READ 文はプログラムにある DATA 文の、始めのデータから読みだします。

note 1 / : 注意すること

DATA の数が少ないと Out of data error になります。

note 2 / : DATA 文にそのまま使えない記号があります。

$$\left. \begin{array}{l} 1 \text{ 12 } 3 4 \\ 2 \text{ 12 } 3 4 \\ 3 \text{ 12 } 3 4 \end{array} \right\} \text{ のようにします。}$$

サンプルプログラムを参考にして、応用して下さい。

日本電話会社の電話番号。デンプン酒にまかれたネーブル酸から目当ての番号をさがすのは大変ですね。名前を打ち込むだけで電話番号がわかる。そんなことができればいいと思いませんか？  
名付けて「コンピュ　タの電話帳」。意外と簡単にできるんです。

```
100 REM -----
```

```
110 REM: デンプン酒
```

```
120 INPUT " シメイヲ イレテタダサイ: "; IN$
```

変数は英字2文字まで有効で、  
3文字目からは無視されます。

```
130 READ SHIS, TEL$
```

ここでは、SHISはSHIS、TEL\$  
はTEL\$として扱われます。

```
140 IF IN$ <> SHIS THEN GOTO 160
```

```
150 PRINT SHIS: " "; TEL$
```

```
160 IF SHIS <> "*" THEN GOTO 130
```

```
170 END
```

```
180 DATA SEGA, 03-742-3171, 名駒, 番号, 名駒, 番号 -----
```

```
999 DATA *, *
```

このプログラムをRUNすると、　　「シメイヲ イレテタダサイ」と、　「コンピュ　タ」が要求してきます。あなたが、たとえば、SEGAと入力すると、コンピュ　タは、あらかじめ入れておいたデータを初めから読んでゆき、さがし出し、名前とその電話番号を表示してくれます。

実行したとき、いったいどのようなことが行なわれたのでしょうか。もう少し詳しく、上のプログラムを分析しましょう。

```

100 REM -----
110 REM デンワチョウ          デンワを1000までと仮定しています。
120 INPUT " シメイヲ イレテウダサイ " ; IN$——名前を入力を要求します。
130 READ SHI$, TEL$          180行の DATA 文のデータをはじめから
                               読んでゆきます。
140 IF IN$ <> SHI$ THEN GOTO 160 ・・・ インプットした名
                               前がデータの中にある名前と一致するか
                               どうか判断しています。
150 PRINT SHI$ : " " : TEL$——データの中に、インプットした名前と一
                               致する名前をみつけたら、名前と電話番
                               号を表示します。
160 IF SHI$ <> " ** " THEN GOTO 130 ・・・ 一致する名前が
                               みつからないと、130行へもどってデータ
                               を読み続けます。
170 END
180 DATA $EGA, 03-743-3171, ...,   あらかじめたくさんの名前
                                     と電話番号をデータとして入れておきます。
999 DATA **, **  ....  データの終わりを示しています。READ 文
                               は、はじめから順にデータを読み続け、
                               終わりのデータ **, ** を読んだところ
                               で、読むのをやめます。

```

★★★ RESTORE (リストアー) ★★★

電話帳のプログラムに、RECORD語句を付け加えて、本本を読んだあとに、プログラムの始めにもどり、名前をきいてきます。

次のプログラムは、デンワチュウのプログラムにRECORD語句を付け加えて、繰り返し使えるようにしたものです。

```

100 REM      - - - - -
110 REM: デンワ チュウ
120 INPUT "シメイファイルアタダサイ: "; IN$
130 RESTORE
140 READ SHI$, TEL$
150 IF IN$ < > SHI$ THEN GOTO 200
160 PRINT
170 PRINT SHI$, " "; TEL$
180 PRINT
190 GOTO 120
200 IF SHI$ < > " ** " THEN 140
210 PRINT
220 GOTO 120
230 END
240 DATA [名前], [番号], [名前], [番号], [名前], [番号]
999 DATA **, **

```

**note 1** このプログラムを実際に電話帳として応用するには、190 行から DATA 文を付けた  
ててください。

|     |      |         |    |
|-----|------|---------|----|
| 190 | DATA | 名前、電話番号 | CR |
| 200 | DATA | 名前、電話番号 | CR |
| ↑   | ↑    | ↑       | ↑  |

の形、或いは

190 DATA 名前、番号、名前、番号 …… と続けて延べる形で入  
力します。

プログラムの中の READ と DATA に注目してください、この二つの命令文が重要な役割を果して  
いるのです。

READ と DATA はいつも組みで使います。DATA 文でたくさんのデータを入れておいて、使いた  
いときに READ 文で読み出します。DATA 文が、いくらプログラムの様式であっても、プログラムの  
流れが READ 文のところにくると、DATA を先に読みます。

気をつけることは、READ 文で使う変数の数が、データより多いと Out of data error になると  
いうことです。ないデータを読むということは不可能ですから。

逆に、データの方が多いときは、エラーにはならず、余ったデータを無視して、先へ進みます。その  
先にまた READ 文があれば、データの続きを読みます。

**note 1** データ文に …, ' ' を使う場合は、' ' で囲みます。

\*\*\* さらに便利に \*\*\*

アノサチュウのプログラム、もうひとつ書いておきます。200行から260行に改題、友達の名前と電話番号を付け加えてから使ってみましょう。

誤作するデータがないとき、「トウロクサレタイマセン」と両端に表示するプログラム

```

110 REM アノサ チュウ
120 INPUT " シメイヲ イレテタダサイ : " ; INS
130 RESTORE
140 READ SHIS, TELIS
150 IF INS <> SHIS THEN GOTO 200
160 PRINT
170 PRINT SHIS : " " ; TELIS
180 PRINT
190 GOTO 120
200 IF SHIS <> " ** " THEN GOTO 140
210 PRINT
220 PRINT " トウロクサレタ イマセン "
230 PRINT
240 GOTO 120
250 END
260 DATA SEGA, 03-742-3171 , 名前, 番号, 名前, 番号
999 DATA ** , **

```

## 配列（もうひとつの実数）

配列は、添字（せえし）つきの変数とも呼ばれます。変数ですから、数値や文字を入れる箱には違い  
ないのです。今までに学んだ変数よりも便利なのです。配列を使うと、一度にたくさんの箱を用意  
することができます。

配列を使う場合は、

図1 

の形式で配列を宣言します。たとえば 図1 A (4) とすると、

|       |
|-------|
| A (0) |
| A (1) |
| A (2) |
| A (3) |
| A (4) |



、5つの変数をひとつひとつ用意するよりも便利なのです。

配列には、一次元配列と二次元配列があります。

## 一次元配列

変数（カラム）の中の数字（行）に対してです

例

```
10 DIM D(5),D$(5)
20 FOR N=0 TO 5
30 INPUT " 名前は？ " ;A$
35 INPUT " とくてんは？ " ;A
40 D$(N)=A$
45 D(N)=A
50 NEXT N
60 FOR N=0 TO 5
70 PRINT D$(N);D(N)
80 NEXT N
```

RUNしたら、6人分の名前と得点を順番  
に入れてください。

## 二次元配列

変数がふたつです。

例1 ゲームの点数表を作ります。3名のゲームの得点を表記しましょう。



| コマンド \ ゲーム | A ゲーム   | B ゲーム   | C ゲーム   |
|------------|---------|---------|---------|
| ゲーム 1      | O(1, 1) | O(1, 2) | D(1, 3) |
| ゲーム 2      | O(2, 1) | O(2, 2) | O(2, 3) |
| ゲーム 3      | O(3, 1) | D(3, 2) | D(3, 3) |

```

10 DIM D(3,3)
20 FOR M=1 TO 3
25 INPUT " コマンド = " ; C$(M)
30 FOR N=1 TO 3
40 INPUT " テンソク = " ; A
50 O(N,M)=A
60 NEXT N
80 NEXT M
90 FOR M=1 TO 3
95 PRINT C$(M); " ; " ;
100 FOR N=1 TO 3
120 PRINT D(N,M);
130 NEXT N
140 PRINT
150 NEXT M

```

DIM O(3,3) の一行で9(3×3) 個の値が用意されます。

例2 九九の表を作りましょう。

```
10 DIM J(9,9)
20 FOR N=1 TO 9
30 FOR M=1 TO 9
40 J(N,M)=N*M
45 IF J(N,M)<10 THEN PRINT " ";
50 PRINT J(N,M);
60 NEXT M
70 PRINT
80 NEXT N
```

図表 J(9,9) で、81(9×9) 個の箱が用意され、

|   |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|
| 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
| 2 | 4  | 6  | 8  | 10 | 12 | 14 | 16 | 18 |
| 3 | 6  | 9  | 12 | 15 | 18 | 21 | 24 | 27 |
| 4 | 8  | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 |
| 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 |
| 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
| 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 |

このような表ができあがります

\*\*\* 配列のとりけし...ERASE(イレース) \*\*\*

一度配列を宣言したら、同じ配列をプログラム内でもう一度使いだすことはできません。そのようなときは、いったん配列をとりけして、もう一度宣言します。

配列を無効にするには ERASE を使います。次の3つのやり方があります。

100 ERASE (プログラム内の配列をすべて無効にします。)

100 ERASE 配列名 , 配列名 , ..... (指定した配列のみ無効にします。)

*note 1 /* 配列を宣言すると、配列を入れるメモリだけ先に確保しますから、あまり大きな配列を宣言すると、プログラムする場所が小さくなります。

*note 2 /* このページは、数字の最大値 100 までにはすでに用意されているので、DIM (100) までの配列は宣言をしなくてもかまいません。その場合は、自動的に宣言されます。しかし、DIM (100) 以上の配列を使うときは、必ずプログラムのはじめに、DIM 配列名(数字) と入力して宣言してください。

## 文字列関数と関数

変数のことを思い出してください。数値の変数と文字の変数の二種類がありましたね。

プログラムに変数は不可欠で、変数という器に数値、あるいは文字を盛り込んで使うことに慣れ込んでいます。

ここでは、変数の器に入れる数値や文字を置く方法を学びましょう。

### 文字列と数値の入れ換え VAL と STR

\*\*\* VAL (バリュー) \*\*\*

VALは、文字列として振われている数値を、数値に変えます。

例1

|                   |                    |
|-------------------|--------------------|
| PRINT VAL ("123") | → 文字列としての123か……    |
| 123               | → 数値の123として表示されました |

*note /* 数値を両端に大括弧したとき、先頭にマイナスのスペースができます  
これは、+ (プラス) 記号が入るところですか、それを省略してあるためです。

## 例2

```

10 A$ = "234"
20 B$ = "222"
30 C$ = A$ + B$
40 C = VAL(A$) + VAL(B$)
50 PRINT C$
60 PRINT C
RUN
23422          - C$ (文字列のたし算の結果)
456             - C  (数値のたし算の結果)

```

\*\*\* STR\$(エス・ティー・オールダラー) \*\*\*

数値を文字列にします。

```

10 A=123
20 B=456
30 C$ = STR$(A) + STR$(B)
40 PRINT C$
RUN
123 456

```

- 数値の123と  
456が……  
- 文字列に変えられ、たし算されました -

## アスキーコードのこと ASC と CHR\$

コンピュータは文字や記号を理解できません。コンピュータ内部ではすべて、アスキーコードと呼ばれる番号で理解しています。(仕組のチャタリング・セクタリング参照)

文字や記号に、`ASCII` 値の交換をするのが `ASC` 文と `CHR` 文です。

### \*\*\* `ASC` (アスキー) \*\*\*

`ASC` 文は、例のように文字を数値に変換します。

? `ASC` ("Z")

90

? `ASC` (" ")

32

? `ASC` ("1")

49

↑ スペース (空白) です。これも文字の1つです。

このように文字を数値にしますから、文字の個数をくらべることができます。

`ASC` と反対の命令が `CHR` 文です。

### \*\*\* `CHR` (キャラクター) \*\*\*

数値を文字記号に変えます。

アスキーコードの 0 から 255 のうち、32 から 255 までの数値に、記号と文字が割りあてられています。0 から 31 まではコントロールコードとよばれ、画面に出力されても文字や記号を表示することはありませんが、何らかの特別な働きをします。

例1( 32 から 255 までのアスキーコードを文字、記号に変換 )

```
10 FOR N=32 TO 255
20 PRINT CHR$(N); " ";
30 NEXT N
```

□ 1 2 3 4 ……

↑ CHR\$(32) はスペース (空白) です。スペースも文字と同じ扱いを受けます。

例2( コントロールコード 17 と 18 の働き )

```
10 SCREEN 2:CLS
20 PRINT CHR$(17) ← 文字を横2倍にする。( SCREEN 2 のときに限ります )
30 PRINT " ABC "
40 PRINT CHR$(18) ← 文字を縦半にする。
50 PRINT " ABC "
```

文字列を扱う LEN, LEFT\$, RIGHT\$, MID\$

\*\*\* LEN ( レンダス ) \*\*\*

LEN は、文字列の長さを調べます。

```
10 A$ = " BASIC "
20 N=LEN(A$)
30 PRINT N
RUN
5
```

A\$ の持つくとも、"BASIC" は何文字であることを調べました。

\*\*\* LEFT\$, RIGHT\$, MID\$ \*\*\*

長い文字列の中から、文字の一部を取り出す関数です。

例 1 LEFT\$ (レフトダラー)

文字列 A\$ の、左から 4 番目までを取り出して、M\$ の中に入れ、画面に表示させます。

```
10 A$ = " コーヒー ココア ミルク "
```

```
20 M$ = LEFT$ (A$ , 4)
```

```
30 PRINT M$          L——( 左から 4 番目までの文字 )
```

```
RUN
```

コーヒー

例 2 RIGHT\$ (ライトダラー)

文字列 A\$ の、右から 3 番目までの文字を取り出して M\$ の中に入れ、画面に表示させます。

```
10 A$ = " コーヒー ココア ミルク "
```

```
20 M$ = RIGHT$ (A$ , 3)
```

```
30 PRINT M$          L——( 右から 3 番目までの文字 )
```

```
RUN
```

ミルク



43 MIC 1 (2, 195-)

文字列の左から6番の文字を起点として3つの文字を取り出して、11番の中に入れ、両端に書き加えます。

```
10 A$ = " コーヒー ココア ミルク "
```

```
20 M$ = MID$(A$, 6, 3)
```

```
30 PRINT M$
```

↑ 取り出す文字数

```
RUN
```

結果

```
ココア
```

問題 5 30N, ABS, HEX E

◆ ◆ ◆ 2014 ( 學 年 ) ◆ ◆ ◆

あふ数の正負符号を知らず。

PRINT SIGN (-)

ある数が正 (例、120 など) のとき  $\Rightarrow$  2 次関数のグラフが x 軸と 2 点で交わる。

ある数が0の上昇  $\Rightarrow$  0

ある数が負（-7, -150 など）のとき  $\theta = -1$ 

3 階層の設置しがありません。

\*\*\* ABS (アブソリュート) \*\*\*

ある数の絶対値を求めます。

絶対値とは、その数字についている符号 (正・負) を取り去った形をいいます。

```
例   PRINT ABS (-5)
      5
```

```
PRINT ABS (8 * (-5))
      30
```

```
PRINT ABS (2 + 3)
      5
```

\*\*\* HEX \$ (ヘキサダラー) \*\*\*

10進数を 16進数に変換します。コンピュータは、10進数よりも 16進数の方が得意です。

```
PRINT HEX $ (255)   この値は  PRINT &HFF
FF                   255
```

<10進数と 16進数>

10進数 ①123456789 10 11 12 13 14 15 16

16進数 ①123456789 A B C D E F 10

16進数では F の次で始まりして 10 になります。

## 音楽をプレイしよう

### PLAY文とSOUND文。

この2つを使うと言われます。

音楽を奏でるにはドレ、……で入力できるPLAY文、効果音や観音を出すのは周波数で入力するSOUND文がよいでしょう。

まずはPLAY文。

### \*\*\* PLAY (プレイ) \*\*\*

PLAY "CEG" [C8]

PLAY文と文字列で、音を認識してくれます。文字列の中には、音高、オクターブ指定など、いろいろな要素が入ります。いろいろな要素って?? これから順々に説明していくとしましょう。

PLAY文で指定できる要素は

- |             |                       |
|-------------|-----------------------|
| ・ 音高        | トレ：ファ…… (もしくはCDEFG……) |
| ・ オクターブ指定   | Qに数字をつける (1～5)        |
| ・ 音符 (音の長さ) | 音高の数字を*つける (0～9)      |
| ・ 休止符       | Rに数字をつける (0～9)        |
| ・ 音の大きさ     | Vに数字をつける (0～15)       |

- ・ テンポ (遅さ)
- ・ 音色
- ・ 和音

## 1. 881 (K278-001)

異業、また、ひらがなのいっすねかで入力します。

そして、アエはアに告げても聞かれない。



|   |    |      |   |   |      |   |   |   |
|---|----|------|---|---|------|---|---|---|
|   | 原典 | F    | V | z | フ    | フ | ウ | リ |
| A | 発子 | C    | D | E | F    | G | A | B |
|   | カタ | F, ト | ウ | z | フ, フ | ソ | ウ | リ |
| リ | カタ | ト, ト | レ | ム | フ, フ | セ | ル | シ |

半角は背高の部に半(ジョーブ)や平(フラット)を付けます。指定した背高のみ有効です。

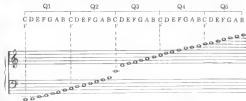
1994



"CRF" Vol. 1, 2, 3, 4

## 2 オクターブの指定

図を参考に、Q1～Q5 のいずれかに指定します。(電源を入れたときは Q3 になっています。)



一度 Q4 などと指定した場合は、次のオクターブ指定があるまでその指定は有効です。

臨時に一度だけ上または下のオクターブの音を指定する時は、(例)のように、音画符号の直前に + (高)、または - (低) をつけます。この指示はその音のみ有効です

(例)



"Q3 ♯5 / Q4 ♯ Q3 ♯"

又は

"Q3 ♯5 / + ♯ -"

### 3. 音の長さ (音符)

| 音符 |  |  |  |  |  |  |  |  |  |  |
|----|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| 記号 | 0                                                                                 | 1                                                                                 | 2                                                                                 | 3                                                                                 | 4                                                                                 | 5                                                                                 | 6                                                                                 | 7                                                                                 | 8                                                                                 | 9                                                                                 |

音を記号に、音高の後ろに数字をつけてください。

(電源を入れたとき、またはリセットしたとき、音の長さは、5になるのが原則です。) 音高と音の長さを指定して音階にします。

たとえば、C3は「ドの 四分音符 (♩)」です。

一度音の長さを決めれば、その後に続く音は同じ長さで演奏されます。

(例)

～♪♪♪♪♪～

```
10 PLAY "C3AGFC5R3
20 PLAY "C1CC3AGFD5R
90 END
```

～♪♪♪♪♪～

```
10 PLAY "C3AGFC5R3
20 PLAY "C1CC3AGFD5R
50 PLAY "E3+C#BAE5R3
60 PLAY "E1EE3+C#BAG5E
90 END
```

ノープとノリットを使っています

note / 記号を、あらかじめC3AG

のように書き変えておくと、入力すると

のように書き変えておくと、入力すると

#### 4. 休止符

|     |    |    |    |    |    |    |    |    |    |    |
|-----|----|----|----|----|----|----|----|----|----|----|
| 休止符 |    |    |    |    |    |    |    |    |    |    |
| 記号  | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 |

Rは休止符の記号です。休止符は音符と同じように使います。

表を参考に、Rの後ろに数字を付けてください。

(例) PLAY "C5RER3G5" **[CR]**

電源を入れたとき、またはリセットしたときは、R5になっています。

さて、今までのデタニッシュを使って、一曲演奏してみませんか。

カエルの唄 — 練習です —

```

200 PLAY "T160:T160"
210 PLAY "Q4CDEFEDCR5:R5RRRRRRR"
220 PLAY "Q4EFGAGFER5:Q1CDEFEDCR5"
230 PLAY "Q4C7CCC:Q1E5FGAGFER5"
240 PLAY "Q4C3CDDDEEFF:Q1C7CCC"
250 PLAY "Q4E5DCR5:Q1C3CDDDEEFF"
260 PLAY "Q4R5RRRRRRR:Q1E5DCR5"
290 GOTO 200

```

かみゆの歌



note 1 / 200 行の T 100 はテンが指定です。

後で説明します。

note 2 / 210 行 ~ 280 行の ♯ は、音を準えるためのものです。転写になっているのは、このためです。〔和音の項 参照〕



## 5 音の大きさ

V0 から V15 まで使って音の大きさを定めます。

(例)

```
10 PLAY "V8 C3AGFC5
20 PLAY "V15 C1CC3AGFD5
90 END
```

V0 OFF (次の音の大きさを定めるまで音がでません。)

?

V15 最大音量

電源を入れたとき、またはリセットしたときは、V12になっています。

## 6 速さ (テンポ)

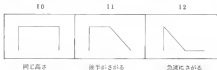
1 分間に四分音符をいくつ使うかを数字で指定します。(T40 ~ T200 の範囲で指定。)

(例)

```
5 PLAY "T40"
10 PLAY "V8 C3AGFC5
20 PLAY "T160 V15 C1CC3AGFD5
90 END
```

## 7. 音色

音色の記号は1です。10から12まで3つの音を使い分けられます。



この各の波型を参考にして音を組み立てましょう。

note /

タイ  2つ以上の同じ高さの音を一つの音符のように演奏します。  
 ♪ 10C311C

スラー  2つ以上の違う高さの音を切れ目なく演奏します。  
 ♪ 10C311E

タイ、スラー、三連音符などは楽器の種類や演奏のテクニックによるものですから、実音をそのまま入力してもきれいに聞えない場合があります。耳で確かめながら修正しましょう。

8. 和音（2つ、或いは3つの音を重ねる）

文字列の中を“:”で区切ると和音が出せます。

(例1) `PLAY "ト:ミ:ソ" CR` （ト、ミの和音）

*note 1 /* 二重和音、三重和音を含む楽譜は、小節ごとに行番号をつけると間違えずに入力できます。

*note 2 /* Q、V、T、Iの指定は各和音それぞれ独立して設定できます。

(例2) `PLAY "T60Q1C5EG:T120Q4C5EGGEC"`

例2では左側は四分音符が3つ、右側は四分音符が6つですね。でも、テンポがT60とT120のように、2倍になっているため、PLAYしたとき、左は2秒に終わります。

和音を“:”で区切った場合、タイミズダを合わせたいときは、最初に1つだけ音符や休止符を入れます。

(例3) `10 PLAY "R:R"`

`20 PLAY "CEG:-E-G-E"`

*note 1 /* “-”をつけると、オクターブ低くなります。

和音を使用した長い曲のタイミングを合わせるには、音符や休符で 1/60 秒の区切に細数が異なるテンポにセットすると良いでしょう。

たとえば、T=90、T=120 など

## PLAY 文 サンプルプログラム

カチューシャ プランテール作曲

楽譜の小節とプログラムの行番号は 10 行ごとに対応しています。

|    |      |              |            |      |  |
|----|------|--------------|------------|------|--|
| 5  | PLAY | "R           | :R         | "R"  |  |
| 10 | PLAY | "F8G3        | :R7        | :R7" |  |
| 20 | PLAY | "A\$8F3      | "R7        | :R7" |  |
| 30 | PLAY | "A\$3A\$GF   | :R7        | :R7" |  |
| 40 | PLAY | "G5C3R3      | :R7        | :R7" |  |
| 50 | PLAY | "G8A\$3      | :R7        | "R7" |  |
| 60 | PLAY | "B\$8G3      | :R7        | :R7" |  |
| 70 | PLAY | "B\$3B\$A\$G | :R7        | :R7" |  |
| 80 | PLAY | "F7          | "R7        | :R7" |  |
| 90 | PLAY | "G4C5F       | :A\$8Q4D\$ | :R7" |  |

和音のスタートタイミングを合わせるため

単音の演奏

|     |      |                             |          |
|-----|------|-----------------------------|----------|
| 100 | PLAY | "10E\$511F3E\$-C5D\$3C-R7"  | 2 重和音の演奏 |
| 110 | PLAY | "10D\$311D\$-D3B\$3B\$ :R5" |          |
| 111 | PLAY | "10C311-B\$3:A\$G :R5"      |          |
| 120 | PLAY | "C5-F :A\$5F :R7"           |          |
| 130 | PLAY | "R3D\$5-B\$3-R3B\$5G3 R7"   |          |
| 140 | PLAY | "C5-A\$3 :A\$5F3-R7"        |          |
| 150 | PLAY | "Q3G3CA\$G:G3CDE-R7"        |          |
| 160 | PLAY | "F7 :F7 :R7"                | 3 重和音の演奏 |
| 170 | PLAY | "D4C5F :A\$5+D\$ :R5G"      |          |
| 180 | PLAY | "10E\$ :+C5 :A\$3"          |          |
| 181 | PLAY | "11F3 :+D\$3 :F5"           |          |
| 182 | PLAY | "E\$ :+C :A\$3"             |          |
| 190 | PLAY | "10D\$ -B\$3 :R3"           |          |
| 191 | PLAY | "11D\$ :B\$ :F3"            |          |
| 192 | PLAY | "10C11-B\$ -A\$G :F5"       |          |
| 200 | PLAY | "11C5-F :A\$5F :F5F"        |          |
| 210 | PLAY | "Q3R3 :R3 :R3"              |          |
| 211 | PLAY | "+D\$5 :B\$5 :F5"           |          |
| 212 | PLAY | "B\$3 :G3 :F3"              |          |
| 220 | PLAY | "+C5A\$3 :A\$5F3:F6R3"      |          |

```



230 PLAY 'GCA$G      :G CDE   R7"
240 PLAY 'F7          :F7       R7"
300 GOTO 5

```

2 重和音の演奏

110, 180, 190, 210行は1行に納まらないので分割してあります

100行の10, 11でスラー()の感じを出します。

110行の10, 11でタイ()の感じを出します。タイは同じ高さの音符を同一の音符のように演奏するので、 を  に置き替えても良いのですか、演奏を聞いて感じの良い方を使います。

*note /* PLAY文の実行は1行ずつ実行されるのではなく、文字列を一度バッファ(メモリーの一部)に取り入れてから実行します。ですから、演奏中にBREAKキーを押しても、止まった行番号と、実際に出ていた音とは一致しません。

## ＜演奏おしまい＞ PLEN (プレイエンド)

プログラムの中でPLAY文を使うと、音楽を演奏した後にプログラムを実行します

しかし、演奏とそれ以外のプログラムを、別々に実行したい場合もありますね。そんなときはこのPLEN文を使います。

|                                                        |                                                         |
|--------------------------------------------------------|---------------------------------------------------------|
| IF PLEN(n)=-1 THEN XXX<br>又は<br>IF PLEN(n) THEN XXX    | nチャンネル (n=0の場合は全チャンネル) の<br>PLAYが終了したら、XXXへジャンプ         |
| IF PLEN(n)=0 THEN XXX<br>又は<br>IF NOT PLEN(n) THEN XXX | nチャンネル (n=0の場合は全チャンネルのど<br>れか一つでも) がPLAY中ならXXXへジャン<br>プ |

note / PLAY文の文字列の中で<sup>①</sup>、<sup>②</sup>では切られた部分をチャンネルといいます  
よから<sup>①</sup>1チャンネル、<sup>②</sup>2チャンネル、<sup>③</sup>3チャンネル<sup>④</sup>です。

(例1)

```
10 PLAY "CDEF:EGEG:+CBAG"  
20 PLAY "CDEF:EGEG:+CBAG"  
30 PLAY "CDEF:EGED"  
40 PLAY "CDEF:EGED"  
45 IF PLEN(3)=-1 THEN 50  
49 GOTO 45  
50 SCREEN 2:CLS  
60 FOR R=10 TO 80 STEP 4  
70 CIRCLE(128,96),R,15  
80 NEXT R
```

(例2)

```
10 PLAY "CDEF:EGEG:+CBAG"  
20 PLAY "CDEF:EGEG:+CBAG"  
30 PLAY "CDEF:EGED"  
40 PLAY "CDEF:EGED"  
45 IF PLEN(3)=0 THEN 45  
50 SCREEN 2:CLS  
60 FOR R=10 TO 80 STEP 4  
70 CIRCLE(128,96),R,15  
80 NEXT R
```

どちらのプログラムも、20行の繰り返しが終わると円を描きます。



# \*\*\* SOUND (サウンド) \*\*\*

これは SOUND 文です。

ゲームに使う効果音や音楽が出せます。トレ：……と音定も出来ます。

同じ プレイ……を唱らすにも、PLAY 文と SOUND 文は違います。PLAY 文では、音高、音の長さなど、いろいろな要素で音を指定しましたね。でも、SOUND 文では、チャンネル、PSGセット値(音の高さ)、音量(音の大きさ)の3要素から音を出します。



では、それぞれについて説明しましょう。

## チャンネル

チャンネルは、0～5のいずれかに指定します。

- 0 : 音を出します。
- 1 : } 音随表にある音と、それ以外の音を出します。
- 2 : } PSGセット値(1～1023までの値)で音が決まります。
- 3 : } 音階は次表を参照してください。

①「ト」②「イ」③「ア」④「エ」⑤「オ」

⑥「ウ」⑦「エ」⑧「オ」⑨「カ」⑩「キ」⑪「ク」⑫「ケ」⑬「コ」⑭「サ」⑮「シ」⑯「ス」⑰「セ」⑱「ソ」⑲「タ」⑳「チ」㉑「ツ」㉒「テ」㉓「ト」㉔「ナ」㉕「ニ」㉖「ノ」㉗「ハ」㉘「ヒ」㉙「フ」㉚「ヘ」㉛「ホ」㉜「マ」㉝「ミ」㉞「モ」㉟「ヤ」㊱「ユ」㊲「ヨ」

ノイズといっても雑音ではないのです。

効果音として使います。

1, 2, 3の3つのチャンネルは、ひとつのチャンネルから4096個の音を作ります。つまり、16個の音まで可能なのです。

例

10 SOUND 1, 214, 15

20 SOUND 2, 179, 15

30 SOUND 3, 143, 15

40 FOR W=0 TO 1000 :NEXT W ①②③④⑤⑥⑦⑧⑨⑩⑪⑫⑬⑭⑮⑯⑰⑱⑲⑳㉑㉒㉓㉔㉕㉖㉗㉘㉙㉚㉛㉜㉝㉞㉟㊱㊲㊳㊴㊵㊶㊷㊸㊹㊺の音です。

50 SOUND 0

PSGセット値（音の高さ）

チャンネル1～3のとき セット値を入れます

音階の表

| Oct. | 音 高   | PSGセット値 | 理論周波数 Hz | 実際の周波数 |
|------|-------|---------|----------|--------|
| Q1   | C     | 858     | 130.8    | 130.8  |
|      | C#, D | 897     | 138.6    | 138.6  |
|      | D     | 742     | 146.8    | 146.8  |
|      | D#, E | 710     | 155.6    | 155.6  |

| Qst. | 音 高    | PSG $\omega_p$ 値 | 理論周波数 Hz | 実際の周波数 |
|------|--------|------------------|----------|--------|
| Q1   | E      | 679              | 164.6    | 164.7  |
|      | F      | 641              | 174.6    | 174.6  |
|      | F#, G# | 605              | 185.0    | 184.9  |
|      | G      | 571              | 196.0    | 195.9  |
|      | G#, A# | 539              | 207.7    | 207.6  |
|      | A      | 506              | 220.0    | 220.2  |
|      | A#, B# | 460              | 233.1    | 233.0  |
|      | B      | 453              | 246.9    | 246.9  |
| Q2   | C      | 426              | 261.6    | 261.4  |
|      | C#, D# | 404              | 277.2    | 276.9  |
|      | D      | 381              | 293.7    | 293.6  |
|      | D#, E# | 360              | 311.1    | 310.7  |
|      | E      | 339              | 329.6    | 329.0  |
|      | F      | 320              | 349.2    | 349.6  |
|      | F#, G# | 302              | 370.0    | 370.4  |
|      | G      | 285              | 392.0    | 392.5  |
|      | G#, A# | 269              | 415.3    | 415.6  |
|      | A      | 254              | 440.0    | 440.4  |
|      | A#, B# | 240              | 466.2    | 466.1  |
|      | B      | 226              | 493.9    | 493.0  |

| Qn. | 音 高    | PSG-1 音 節 | 理論周波数 Hz | 實際中心周波数 |
|-----|--------|-----------|----------|---------|
| Q3  | C      | 214       | 523.3    | 522.7   |
|     | C#, D♭ | 202       | 554.4    | 553.8   |
|     | D      | 186       | 587.3    | 586.7   |
|     | D#, E♭ | 180       | 622.3    | 621.4   |
|     | E      | 170       | 659.3    | 658.0   |
|     | F      | 160       | 698.5    | 699.1   |
|     | F#, G♭ | 151       | 740.0    | 740.8   |
|     | G      | 143       | 784.0    | 782.2   |
|     | G#, A♭ | 138       | 830.8    | 828.8   |
|     | A      | 127       | 880.0    | 880.8   |
|     | A#, B♭ | 120       | 932.3    | 932.2   |
|     | B      | 113       | 987.8    | 989.9   |
| Q4  | C      | 107       | 1046.5   | 1045.4  |
|     | C#, D♭ | 101       | 1108.7   | 1107.5  |
|     | D      | 95        | 1174.7   | 1177.5  |
|     | D#, E♭ | 90        | 1244.5   | 1242.9  |
|     | E      | 85        | 1318.5   | 1318.0  |
|     | F      | 80        | 1396.9   | 1398.2  |
|     | F#, G♭ | 78        | 1480.0   | 1471.9  |
|     | G      | 71        | 1568.0   | 1575.5  |

| OctL | 音 高     | PSGコード値 | 理論周波数 Hz | 実際の周波数 |
|------|---------|---------|----------|--------|
| Q4   | G# , A3 | 87      | 1661.2   | 1669.6 |
|      | A       | 84      | 1740.0   | 1747.6 |
|      | A# , B3 | 80      | 1864.7   | 1864.4 |
|      | B       | 77      | 1975.5   | 1982.5 |
| Q5   | C       | 63      | 2093.0   | 2110.6 |
|      | C# , D4 | 60      | 2217.5   | 2237.2 |
|      | D       | 58      | 2349.3   | 2320.4 |
|      | D# , E4 | 55      | 2489.0   | 2485.6 |
|      | E       | 52      | 2637.0   | 2662.4 |
|      | F       | 49      | 2793.8   | 2796.5 |
|      | F# , G4 | 46      | 2960.0   | 2943.7 |
|      | G       | 43      | 3136.0   | 3107.2 |
|      | G# , A4 | 40      | 3322.4   | 3290.0 |
|      | A       | 37      | 3520.0   | 3495.7 |
|      | A# , B4 | 34      | 3729.3   | 3728.7 |
|      | B       | 31      | 3951.1   | 3995.0 |

↓  
等分平均律による音高

note 1 / PLAY文ではトレ：……やCD音……で入力しますが、SOUND文ではこの表と照らしながら入力しなくてはなりません。

note 2 / 音階表にある PSG キット値以外の数値を入力すると、トとレの中間の音など、トレ：ファ……以外の音が出ます。

(PSG キット値は周波数をもとにした値で、1～1023の範囲の整数です。)

```
100 FOR N=1023 TO 1 STEP -1
110 SOUND 1,N,15
120 PRINT N
130 NEXT N
140 SOUND 0
```

1～1023のすべての音が出ます。)

チャンネルが4重たれるとき — 0, 1, 2, 3のいずれかに指定します。

|         |                                |       |
|---------|--------------------------------|-------|
| キット値 0. | } 三段階の、すでに決定している<br>音の高さになります。 | 512Hz |
| 1:      |                                | 256Hz |
| 2:      |                                | 128Hz |

キット値 3: チャンネル 3 で指定する音の高さになります。

```
(例) 300 SOUND 3, 192, 1 : SOUND 4, 3, 15
335 FOR W=0 TO 500: NEXT W
350 SOUND 0
```

0-15 の数字で指定します。

0: 音を消す

1: 最小の音量

↓

15: 最大の音量

*note* / SOUND 文を使ったら必ず SOUND 0 で音を消します。

サンプルプログラムを実行してみましょう。ドレミファソ…と音が鳴りますよ。

サンプルプログラム (1)

```

10 FOR N=1 TO 30
20 READ A
30 SOUND 1,A,15
35 FOR W=0 TO 200:NEXT W
40 NEXT N
50 SOUND 0
100 DATA 856,782,679,841,671,508,453
110 DATA 428,381,339,320,286,264,226
120 DATA 214,190,170,160,143,127,113
130 DATA 107,96,85,80,71,64,57
140 DATA 53,48,42,40,35,32,28,26

```

次は、図音——

Figure 8. Musical (2)

```
200 FOR N=0 TO 3  
210 SOUND 4,N,15  
220 FOR W=0 TO 500 : NEXT W  
240 NEXT N  
250 SOUND 0
```



\*\*\* BEEP \*\*\*

PLAY 文、SOUND 文の前に、BEEP があります。短い音を出します。

BEEP      ビッと音が鳴る。  
BEEP 0    BEEP 音をとめる。  
BEEP 1    ヒーと鳴り続ける。  
BEEP 2    ビボビボと鳴る。

ゲームの効果音などに使いましょう。

(例)

```
10 DIM A$(12)
20 FOR N=0 TO 12
30 READ A$(N)
40 PRINT A$(N);
50 BEEP
60 NEXT N
70 DATA H,O,M,E," ",C,O,M,P,U,T,E,R
RUN
HOME COMPUTER
```

Ready

## グラフィックス

むずかしい技術はいりません。ページレイアウトに使う命令が「`SCREEN`」が覚えやすければいいのです。  
さあ、トライしましょう。

### SCREEN

#### ★★★ 画面のこと ★★★

画面にはテキスト画面とグラフィック画面の二種類あります。オープンアップからホームページに入った時の画面は、テキスト画面といい、プログラムを書くのに使います。(前章まではずっと、テキスト画面を使っていたのです。)

これからグラフィックをやろうという皆さんは、グラフィック画面という専用の画面に切り替えてはなりません。というのは、グラフィック専用の命令文は、それ専用の画面にしか使えないからなのです。

では、どのようにして画面を選択し指定すればよいのでしょうか。

#### ★★★ SCREEN ★★★

SCREEN 文は次のように使います

—テキスト画面を指定するとよ

SCREEN 1

グラフィック画面を指定するとき

SCREEN 2,1 (画面を切り替えたとき軸がメセリの方に一度移ります。ですからSHIFT+BREAKで画面を再度切替えても軸はもどります。)

SCREEN 2,0 (画面を切り替えたとき、軸は動いたままです。)

note1 / SCREEN 2,1 の1は省略可能です。SCREEN 2と入力してもかまいません。  
しかし、SCREEN 2,0を指定した後は、SCREEN 2,1と、省略せずに入力して下さい。

note2 / テキスト画面からグラフィック画面へ切り替えるときには **SHIFT** キーと **↑/BREAK** キーを、グラフィック画面からテキスト画面へ切り替えるときには **↓/BREAK** キーのみ押してください。

## LINE

線を引きます。  
グラフィック画面は、



このように、横 (X方向) に 0 から 255、縦 (Y方向) に 0 から 255 の座標を持っています。線を引くときは、線のはじめとおわりを座標で指定します。

例①

```
10 SCREEN2:CLS
20 LINE(50,60)-(150,120),6
      ↑           ↑           ↑
      線のはじまり 線のおわり 色の指定
      (x1, y1)   (x2, y2)
30 GOTO 30
```



*note1 /* 色の指定については、COLOR の例を参照してください。

*note2 /* 30行の GOTO 30 は、画面が消えないようにしているのです。「無限ループをつくる」といいます。

*note3 /* 座標の座標を省略すると前に書いた座標から線を引きます (次のプログラムの 30行参照)

```
10 SCREEN2:CLS  
20 LINE(50,50)-(150,50),8  
30 LINE-(50,150),8
```



鍵を置き続けると、テキスト画面にもどってしまいます。グラフィック画面にするには、SHIFTキーを押しながらBREAKキーを打ちます。グラフィック画面からテキスト画面にもどるには、BREAKキーを打ちます。2秒ぐらいでテキスト画面にもどります。

もう一度例①のプログラムをRUNしてもいい。  
ななめの線が引けましたね。



このプログラムに、少し手を加えましょう。例②のように、20行を変えてください。

例②

```
10 SCREEN 2  
20 LINE (50,50)-(150,120),B  
30 GOTO 30
```

↑ Bを加えました

実行すると、



こんなふうに、

LINE 文で引いた線を対角線にした BOX が描けました。このように BOX (箱) を描くときには、LINE 文で対角線を指定して、色指定の後に B をつけます。

さらに手を加えましょう。例③のように、20行を変えてください。

# 例③

```
10 SCREEN 2
20 LINE(50,50)-(150,120),8,BF
30 GOTO 30
```

↑  
一画の中を塗ります。

実行すると、



こんなふうに

画に色がつきました。

note1 / : BFはBox Fillを意味します。

note2 / : LINE文を使うときは、必ず、SCREENでグラフィック画面にしてください。

LINE文で、直線を使うと面白い使い方ができます。

## LINE文の応用

```
10 SCREEN 2:CLS
20 Y=0:C=2 ..... Cは色です。0よりから始めます。
```

```

30 FOR X=0 TO 240 STEP 15
40 LINE(X,Y)-(X+15,Y+15),C,BF
50 C=C+1
60 IF C>15 THEN C=2
70 NEXT X
80 Y=Y+15
90 IF Y>160 THEN END
100 GOTO 30

```

15 図内の大きさです。  
 (15を24に変えると?)  
 色を変えます。  
 色を、ドットにもどします。  
 一段下に四角を描きます。(15を24に変えると?)  
 一番下で終わりにします。

## CIRCLE

弧を描いたら、こんどは円です。

次のように入力します。

|     |                                   |     |     |     |     |     |     |
|-----|-----------------------------------|-----|-----|-----|-----|-----|-----|
| (例) | (1)                               | (2) | (3) | (4) | (5) | (6) | (7) |
|     | 座標                                | 半径  | 色   | 比率  | 始点  | 終点  |     |
|     | CIRCLE (125,95),50,5,100,0,100,BF |     |     |     |     |     |     |

例を詳しくみていきましょう。

- (1) 座標 ----- 円の中心点です。
- (2) 半径 ----- 文字通り、半径です。
- (3) 色 ----- 色の指定。COLOR の項を参照してください。



(4) 比率 …… 真円、よこ長のた円、たて長のた円のいずれかに指定します。

比率 100



真円

比率 100 より小さい



よこ長のた円

比率 100 より大きい



たて長のた円

(5) 始点 …… 書き始めの位置を示します。

(6) 終点 …… 書き終わりの位置を示します。



始点 0  
終点 100 の場合



始点 0  
終点 75 の場合

(7) BP を指定しないと …… 円弧のみ描きます。



BF を指定すると



色で指定した色でぬりつぶします。



B を指定すると



川岡だけでなく、物丸と中心、真丸と中心をむすぶ線も描きます。



CIRCLE 文を使ったプログラムを、ひとつあげておきます。ためしてみてください。

```
10 SCREEN 2,1:CLS
20 FOR R=10 TO 50 STEP 10 (半径が 10 ずつ増えます。)
30 CIRCLE(125,95),R,B,100,0,100
40 NEXT R
50 GOTO 50
```

どうですか。



こんなふうに描けましたか？

note1 / CIRCLE 文で描いた円を消すときは、下地と同じ色で円を描きます。

note2 / CIRCLE 文を使うときは、必ずグラフィック画面にしてください。

## PSET

点を打ちましょう。

一般に、

座標 色  
PSET(x, y), 4

この形で入力します。

次のプログラム、ためしてください。

```
10 SCREEN 2, 1:CLS
20 FOR X=0 TO 240 STEP 20
30 FOR Y=0 TO 190 STEP 20
40 PSET(X, Y)
50 NEXT Y
60 NEXT X
70 GOTO 70
```

こんなふうに、



文字がいろいろ並びましたが、

*note /* : PRINT 文を使うときは、必ずグラフィック画面にしてください。

## PRINT

文字を表示します。使い方はテキスト画面のときと同じです。

## CURSOR

グラフィック画面で、文字の表示位置を指定するときも、CURSOR 文を使います。  
グラフィック画面も座標をもっていますから (LINE の項目の図参照)

**CURSOR x , y**

のように、座標で表示したい位置を指定してください。

*note /* : グラフィック画面の座標の数値はテキスト画面の座標の数値と違いますから  
注意してください。

## COLOR と PAINT

コンピュータ・グラフィックで、

線が引けました。

箱（四角）も円も描けました。

点もいっぱい打てました。

さて、次は色ぬりです。

### ★★★ COLOR ★★★

＜テキスト画面＞

テキスト画面で

```
10 COLOR 1,7,15
```

```
20 PRINT "ABC"
```

と入力し、実行してください。

ABC という文字は黒、画面は水色、画面の周囲（バックドロップ、画面を切り替えても変わらない部分）は白になったでしょう。



10 行をみてください。

① ② ③  
COLOR 1, 7, 15

① は文字の色、② は画面の色、③ はバックドロップの色の指定です。それぞれの数字が、色をあらわしています。(下の表を参照)

| 色番号 | 色     | 色番号 | 色       | 色番号 | 色     |
|-----|-------|-----|---------|-----|-------|
| 0   | 透 明   | 6   | こ い 赤   | 12  | こ い 緑 |
| 1   | 黒     | 7   | 水色(メアン) | 13  | マゼンダ  |
| 2   | ＊ 緑   | 8   | 赤       | 14  | 灰     |
| 3   | うすい緑  | 9   | うすい赤    | 15  | 白     |
| 4   | こ い 青 | 10  | こ い 黄   |     |       |
| 5   | うすい青  | 11  | うすい黄    |     |       |

note1 / : 0 (透明) に指定すると、バックドロップの色がでます。

note2 / : COLOR 文の一部を省略することも可能です。

文字の色指定省略 ----- COLOR , 7, 15

画面 \* ----- COLOR 1, , 15

バックドロップ \* ----- COLOR 1, 7

文字の色だけ指定 ----- COLOR 1

画面 \* ----- COLOR , 7

バックドロップ \* ----- COLOR , , 15

## ＜グラフィック画面＞

グラフィック画面で色を変える方法は、テキスト画面と少し違います。テキスト画面では

COLOR ①,②,③

と入力して、文字の色や地の色を設定しましたが、グラフィック画面では次のようにします。

グラフィック画面の文字の色を変えたいときは、PRINT 文の前に COLOR 文で色を設定します。

```
10 SCREEN2:CLS
20 COLOR 8
30 PRINT "ABC"
```

PRINT 文の前に COLOR 文を置くので、ひとつひとつの文字の色を変えることも可能なわけです。たとえば、

```
10 SCREEN2:CLS
20 COLOR 5
30 PRINT "A";
40 COLOR 15
50 PRINT "B";
60 COLOR 2
70 PRINT "C";
```

このようにすると、Aは青、Bは白、Cは緑で、ABCと表示されます。部分的に文字の色を変え

ることは、テキスト画面ではできません。そこが入きな違いです。

地の色を変えるには、CLS の前に、COLOR 文を使います。

```
10 SCREEN 2:COLOR , 3:CLS
```

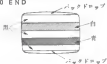
とすると、COLOR 文で指定した色 (3 = 緑) で画面をクリアして (CLS 文)、画面は緑色になります。

note 1 : グラフ・アップ画面の地の色を変えるときは CLS をおれずに、

地の色は、LINE 文の BF でも変わります。

LINE 文を使うと、(LINE 文の項参照)

```
10 SCREEN 2:CLS
20 LINE (0, 40)-(255, 90), 15, BF
30 LINE (0, 120)-(255, 160), 5, BF
40 END
```





こんなふうに画面の一部の色を変えることができます。

部分的に変えることができるかどうか、それがテキスト画面とグラフィック画面の違いのひとつですね。

次のプログラムをためしてみましょう。

```
10 SCREEN 2:CLS
20 FOR C=2 TO 15
30 COLOR C
40 PRINT "■";
50 NEXT C
```

NOTE / : **GRAPH** キーを押したあと V のキーをおすと ■が表示されます。

■ が1個ずつ色が変化っていくでしょう。

こんなことができるのも、グラフィック画面ならではの。

### \*\*\* PAINT \*\*\*

PAINT は、



図のような、LINE 文や CIRCLE 文などで引いた線にかこまれたところを塗ります。一般に、次のように入力します。

P A I N T (x, y) , 色

↓  
斜線の部分（塗りたい部分）にある点のうち、ひとつの座標を指定。

その際、PAINT 文で指定する色と、LINE 文、CIRCLE 文などで引いた線の色は同じでなくてはなりません。違う色が交差していると、そこから色がもれてしまいます。また、ほんのわずかでも、すき間があれば、そこから色がもれてしまいます。注意しましょう。

それでは、このプログラム、ためしてください。

```
10 SCREEN 2:CLS
20 LINE (36, 48)-(200, 144), 5, B ..... 四角を描いて
30 CIRCLE (128, 96), 60, 5 ..... 丸を描きます
40 PAINT (40, 80), 5 ..... 色を塗ると
50 GOTO 50
```

こんなふうに



なったでしょ。

## SPRITE, PATTERN, MAG

### \*\*\* SPRITE と PATTERN \*\*\*

自分がつくったキャラクターが、画面の中を動きまわったなら・・・。そんな願いをかなえてくれるのがスプライト機能。グラフィック画面上にキャラクターをつくり、それを動かす機能です。

PATTERN 文でキャラクターをつくります。SPRITE 文でキャラクターを動かします。

具体的な説明の前に、まずはプログラムを入力し、実行してください。

```
10 SCREEN 2:CLS
20 X=0 : Y=0
30 PATTERN $#100,"FF00FF00FF00FF00"
40 SPRITE 0,(X,Y),100,0
50 X=X+1
60 IF X=256 THEN X=0
70 GOTO 40
RUN
```

すると、



こんなふうに

図本の絵のキャラクターの面をよこごっていきますね。  
どうして、こうなるのでしょうか。

# ＜PATTERN 文 — キャラクターをつくる —＞

プログラムの 30 行を見てください。

P A T T E R N   S#   1 6 0 , '   F F 0 0 F F 0 0 F F 0 0 F F 0 0 '

①   ②   ③

① PATTERN 文は、グラフィック画面、テキスト画面のどちらの画面でも使うことができ、前者の場合は S#、後者の場合は C# と入力します。このプログラムでは S#。S# と指定すると、PATTERN は SPRITE 文で使うことができます。

② の番号は、スプライト名称といいます。つくったキャラクターにつける番号です。キャラクターは全部で 256 個つくることができます。

note1 / /   番号は、16 進数でもかまいません。

例 P A T T E R N   C# & H 3 0

note2 / /   ヘーミングで使うパターン番号と、「パターンへんこう」のパターンでい  
きの番号とは関連があります。(P. 166 参照)

③ は何でしょう？ アルファベット文字が混じっていますが、実は、16 進数であらわした、  
れっきとした数字なのです。

キャラクターを作るときは、「パターンへんこう」の画面で、64x64 のマス目で描きます。こ  
のマス目のひとつ分を 1 ドットと呼びます。ドットをぬりつぶして、キャラクターを描いていきます。



それを左右4ビットずつに分け、の部分に1、の部分をもととして2進数の(0と1の2種類だけの)数字に直します。



| 左       | 右       |
|---------|---------|
| 1 1 1 1 | 1 1 1 1 |
| 0 0 0 0 | 0 0 0 0 |
| 1 1 1 1 | 1 1 1 1 |
| 0 0 0 0 | 0 0 0 0 |
| 1 1 1 1 | 1 1 1 1 |
| 0 0 0 0 | 0 0 0 0 |
| 1 1 1 1 | 1 1 1 1 |
| 0 0 0 0 | 0 0 0 0 |

さらに、それを16進数に変え、(表を参考にしてください。)

変換表

| 左    | 右    | 左 | 右 |
|------|------|---|---|
| 1111 | 1111 | F | F |
| 0000 | 0000 | 0 | 0 |
| 1111 | 1111 | F | F |
| 0000 | 0000 | 0 | 0 |
| 1111 | 1111 | F | F |
| 0000 | 0000 | 0 | 0 |
| 1111 | 1111 | F | F |
| 0000 | 0000 | 0 | 0 |

⇒

| 16 進数    | 2 進数       | 16 進数    |
|----------|------------|----------|
| 0        | 0000       | 0        |
| 1        | 0001       | 1        |
| 2        | 0010 (桁上り) | 2        |
| 3        | 0011       | 3        |
| 4        | 0100       | 4        |
| 5        | 0101       | 5        |
| 6        | 0110       | 6        |
| 7        | 0111       | 7        |
| 8        | 1000       | 8        |
| 9        | 1001       | 9        |
| 10 (桁上り) | 1010       | A        |
| 11       | 1011       | B        |
| 12       | 1100       | C        |
| 13       | 1101       | D        |
| 14       | 1110       | E        |
| 15       | 1111       | F        |
| 16       | 10000      | 10 (桁上り) |

左右左右・・・と繰り返したものが①の「FF00FF00FF00FF00」なのです。「パターンのへんこう」画面で働き、16 進数に変えたデータ (FF00・・・) を、PATTERN Str 160, "   " に入れて実行します。どのようなキャラクターも、このような手順で作ります。その際、16 進数であらわした文字列の両側に " " (ダブルクォーテーション) と、文字列の前に , (コンマ) を必ずつけてください。

# ＜SPRITE 文 — キャラクターを動かす —＞

次のプログラムの 40 行を見てください。

```
40 SPRITE ①, (X, Y), 160, ②
           ①      ②      ③      ④
           画面号 座標   スプライト 色
                        名 称
```

① はスプライト名称。動かしたいキャラクターです。このプログラムの場合、30 行の PATTERN 文でつくったキャラクター (160 番) を呼びだしています。

② で、そのキャラクターの色を決めます。この場合は赤 (赤) です。

PATTERN でつくったキャラクターを呼び出して、色をつけました。動かすのは ② の座標指定です。座定のしかたによって、キャラクターはあちこちへ動きます。

|              |             |
|--------------|-------------|
| キャラクターを右に動かす | $X = X + 1$ |
| “ 左に動かす      | $X = X - 1$ |
| “ 下に動かす      | $Y = Y + 1$ |
| “ 上に動かす      | $Y = Y - 1$ |

このプログラムの場合は、 $X = 0$ 、 $Y = 90$  (20 行)、 $X = X + 1$  (30 行) と入力しているので、四角形のキャラクターは左端 ( $X = 0$ ) から右へと動いていくのです。

note / ・ スプライトが画面の端まで行ったらどうなるでしょう。右に動いていった場合はエラーにならず反対側から出て来ますが、それ以外の方向の画面の端では

エラーになります。

そのため、サブタイトルが両面の間で止まるように制御をつけます。「リセットをつける」と言います。

60 行の IF 文が、右側のリセットです。

左側のリセット

IF X=0 THEN X=255      もとにもどります。

または、

IF X=0 THEN X=X+1      同じ所で止まっています。

上や下のリセットも同様です。X の代わりに Y をあてはめて考えてください。

さて、①の要素号について少し説明します。

サブタイトル面は 32 枚あり、0、1、2、3、... 31 と番号が振られています。



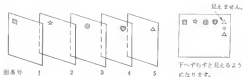


キャラクターを描くときには、スプライト面 32 枚のうちのどれか 1 枚を選びます。何番の面に描くか指定するのが ① の面番号です。

スプライトには優先順位があり、番号の小さい面に描かれたキャラクターほど、優先順位が高いのです。いくつかのキャラクターが画面上を動いていると、キャラクターがかさなりあう場合もでてきますね。そのときに、優先順位の低い方のキャラクターは、優先順位の高い方のキャラクターにかわれてしまうのです。交差した場合にどちらが前にきたらよいか、ということを考えながら、スプライト面番号の指定をすることが大切でしょう。

#### mode 1 : スプライトの制限

キャラクターは、水平線上にもつまで並べられます。もし、5 つ、水平線上にキャラクターを並べたら、面番号の大きい（優先順位の低い）キャラクターは見えなくなります。



## < SCORN >

スプライトを使ってゲームを作る場合、スプライトがぶつかったかどうか判定できればなりません。この判定は次のようにします。

```
SCORN  RUN  -1  衝突した。  
        0     衝突しない。
```

```
IF SCORN THEN ~
```

又は、

```
IF SCORN=-1 THEN ~
```

THEN の後には、プログラムに応じた命令をつけます。

```
THEN BEEP  
THEN GOSUB 行番号  
THEN PLAY 　　　　　　　　　　など
```

面白いアイデアを考えましょう。

サンプルプログラム

```
10 SCREEN:CLS  
20 PATTERN $160," FF00FF00FF00FF00 "  
30 PATTERN $164," 103C7EFFFF7E3C10 "  
50 X=0:XX=255:Y=100:E=1
```

```

60 SPRITE 10, (X, Y), 160, 8
70 SPRITE 11, (XX, Y), 164, 8
80 X=X+E:XX=XX-E
90 IF SC0IN=-1 THEN GOTO 200
100 GOTO 60
200 BEEP:BEEP
210 CLS
220 GOTO 50

```

### ＜BASICと「パターンのへんこう」＞

BASICのパターン番号と「パターンのへんこう」のパターンでいざの番号は関連があるということ  
は、すでに触れました。関連があるとはどういうことなのでしょう。

仕組の「BASICのSP PATTERNとパターンでいざまとの関係」を見てください。

| BASIC | パターンでいざ |
|-------|---------|
| SP 0  | 0       |
| 1     |         |
| 2     |         |
| 3     |         |
| 4     | 1       |
| 5     |         |

これは、図に書くと、

|               |               |
|---------------|---------------|
| BASIC<br>S# 0 | BASIC<br>S# 2 |
| BASIC<br>S# 1 | BASIC<br>S# 3 |


パターンていど 0

|               |               |
|---------------|---------------|
| BASIC<br>S# 4 | BASIC<br>S# 6 |
| BASIC<br>S# 5 | BASIC<br>S# 7 |

パターンていど 1

のような関係になります。BASIC の PATTERN 文でキ+ラチキをつくって RUN した場合、BASIC の S#0 はパターンていど 0 の左上の部分に、BASIC の S#1 はパターンていど 0 の右下の部分に影響を与えます。



たとえば、きき推どの  は、S#169 ですから、RUN した場合、パターンていど 40 の左上の部分に影響を与えます。ために、「パターンへんこう」画面にして、40 を出してご覧下さい。



このように、左上の部分に S#169 がはいっていることがわかるでしょう。

パターンでいう 0～39 (BASIC のパターン番号 0～159) は、ゲームのキャラクターとして使われています。ですから、BASIC では、39 160 からあとの番号を使った方がよいでしょう。(ゲームのキャラクターの形が変わってしまいますから。) もし、ゲームのキャラクターを変更して保存しておくのでしたら、BASIC で 8 までの 00～39 にパターンを描いてテープに取っておきましょう。

### <テキスト画面の PATTERN 文 (C# 指定の場合)>

ふつうは、A のキーを押せば A という文字が、/ のキーを押せば / という記号が画面に表示されます。でも、キーを押したとき、別の文字や記号が表示されるようにすることだって、PATTERN を使えばできるのです。次のプログラムを入力してみてください。

```
1000 PATTERN C# 55, "708898A8C8887000"
```

RUN して、A のキーを押すと \_\_\_\_。A ではなく 0 が画面に表示されました。

"708898A8C8887000" は、16 点数で、0 の形をあらわしています。



左

|   |   |   |   |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 |

2 点数

黒いマス = 1, 白いマス = 0

右

|   |   |
|---|---|
| 7 | 0 |
| 8 | 8 |
| 9 | 8 |
| A | 8 |
| C | 8 |
| 8 | 8 |
| 7 | 0 |
| 0 | 0 |

16 点数

65 は、A のキャラクタコード(=0)に、“既定値”のようなもので「65番はA」と決まっています。ところが、PATTERN文を使って、「65番は0」と変更してしまったのでAのキーを押しても0が表示されるのです。

一般に、テキスト画面の PATTERN 文は、

P A T T E R N C# キャラクタコード, “文字列式”

の形をとります。

キャラクタコードは、付録を参考にしてください。

### ★★★ MAG ★★★

MAG は SPRITE で動かすキャラクタの大きさを定める命令です。P.144 のプログラムに、MAG2 という1行をつけ加えてみましょう。 40 MAG 2

(例) 10 SCREEN 2:CLS  
20 X=120: Y=90  
30 MAG 2  
40 PATTERN \$F160, 'FFFFFF00FF00FF00'  
50 SPRITE 0, (X, Y), 160, 3

すると、



こんなふうに

マスとマスが2倍、面積が4倍の  
大きさになります。

MAG には、MAG0、MAG1、MAG2、MAG3がありますが、それぞれは、そのような働きをするのでしょいか。

PATTERN 文を使って、SF160、SF161、SF162、SF163 の4つのパターンをつくったとします。



SF160



SF161



SF162



SF163

そして、

たとえば

SPRITE 2, (X, Y), 160, 0

というように、SPRITE 命令をするとき、

① MAG0 に指定すると

PATTERN 文でつくった S#160 が、そのまま (8) × 8 (ピクセル) の大きさ) であらわれます。

② MAG1 に指定すると

S#160, S#161, S#162, S#163 の4つのパターンを組み合わせた形であらわれます。

③ MAG2 に指定すると

S#160 が、MAG0 の2倍の大きさであらわれます。

④ MAG3 に指定すると

S#160～163 を組みあわせた形が、MAG1 の2倍の大きさであらわれます。





— プログラムを起動するには —

(第一分限-第二分限-第三分限-第四分限)

グラフィックスの重いプログラムを実行した場合、実行後、または 2D/3D 実行モードを押したとき、オーバーフローエラーになることがあります。これは、VRAM のデータがメインメモリーに退避し、ようとしても、メインメモリー内のプログラムが大きくて退避されなかったためです。このような場合は、メインメモリーのプログラムを削除して、VRAM のデータを退避させます。

プログラムが長すぎて、オーバーフローが出た場合は、以下のプログラムの100行以下を、そのプログラムに追加してください。

[illegible]

ア、イ、ロ、ハ、ニ、ホ、ヘ、セ、  
エ、オ、カ、キ、ク、ケ、コ、サ、  
シ、ス、セ、ソ、タ、チ、ツ、テ、ト、

プログラムを調べる  
プログラムです

[illegible]

実行すると、グラフィックスを描いた後、CALL 文のある行で BREAK (プログラムが止まること) します。そうしたら、SHIFT + BREAK でグラフィックに切替えて絵を確認しましょう。(絵は VRAM に転送されます。) 確認した後、BREAK キーでテキスト画面にもとし、SAVEV "ファイル名" で古セットアップにグラフィックスをセーブしてください。

100-120行のプログラムは、実行後、メモリ内のプログラムをすべて消してしまいます。ですから、実行する前に、かならずメモリ内のプログラムをテープにセーブしておいてください。

## SAVEV , VERIFYV , LOADV

テキスト画面のプログラムを保存することができるよう、グラフィック画面に描いたものやスプライトパターンも、やはりセクタ（？）に保存できます。すなわち、SAVE、LOADと同じです。

### \*\*\* SAVEV \*\*\*

グラフィック画、及びスプライトパターンをセクタテープへ保存します。

SAVEV "ファイル名"

のように入力します。

### \*\*\* VERIFYV \*\*\*

セクタテープに、正確にSAVEされたか確かめます。

VERIFYV "ファイル名"

のように入力します。

### \*\*\* LOADV \*\*\*

セクタテープに保存されているグラフィック画、及びスプライトパターンを、コンピュータのメモリの中へ取り込みます。

LOADV "ファイル名"

のように入力します。

**LOADV** (ファイル名指定無し)

の場合は、最初ファイルを読み込みます。

SAVE、VERIFY、LOAD の後 (P. 52) も参照してください。

LOADV で Loading end が画面に出たら、SHIFT + BREAK でメニューを画面にします。終了させられます。

## INKEY\$とRND

ゲーム作りに不可欠な2つの命令を紹介します。  
まずはINKEY\$。

### \*\*\* INKEY\$ (インキーダラー) - すばやい判断 \*\*\*

キーボードが押されたかどうかをすぐに判断します。

たとえば

```
10 PRINT INKEY$ : GOTO 10
```

をRUNして、キーをとれてもいっから押してください。押したとき画面が表示されます。これは、INKEY\$を、GOTO 10を使って無限ループ(ぐるぐる回して、そープログラム)の中に入れたためです。このようにすると、INKEY\$は、いつキーが押されるか、常に監視しています。そして、キーが押されたら、すぐにその文字も教えてくれるのです。ただし、INKEY\$で受けつけるのは、英数字、10のノーマルとソフト、それにコントロールキー+長いです。





次のプログラムを見てください。



```
5 REM キー コントロール
10 CLS
20 X=15:Y=10:E=1
100 CURSOR X,Y:PRINT "A"
105 Z$=INKEY$
110 IF Z$=CHR$(28)THEN E=1:GOSUB 200
120 IF Z$=CHR$(29)THEN E=-1:GOSUB 200
```

```

130 IF Z$=CHR$(30) THEN E=-1:GOSUB 300
140 IF Z$=CHR$(31) THEN E=1:GOSUB 300
170 GOTO 100
200 CURSOR X,Y:PRINT " "
210 X=X+E:IF X<=0 OR X>28 THEN X=X-E
220 RETURN
300 CURSOR X,Y:PRINT " "
310 Y=Y+E:IF Y<=0 OR Y>=22 THEN Y=Y-E
320 RETURN

```

105 行に INKEY\$ が使われています。CHR\$(28),CHR\$(29),CHR\$(30),CHR\$(31) はコントロールコード(「文字関数と関数」の CHR\$ の項、或いは付録の表を参照してください)で、それぞれ     の、4つのカーソルキーです。

 のキーを押すと、INKEY\$ はすばやく判断して、200 行へとびます。つまり A の文字は右へ動きます。 を押すと、またすぐに判断して 300 行へとび、A の文字は下へ動きます。A の文字は、カーソルキーによって、上下左右へ動くわけです。

これに応用すれば、ゲームの中で、キャラクターを、キーコントロールで動かすことができます。

ゲームに便利なもうひとつの命令は、RND です。

\*\*\* RND(ランダム) - ランダム性 - \*\*\*

ランダムは乱数のサイコロを振ったとき、何が出るかわからないような、規則のない数のことです。

```

10 FOR R=1 TO 20
20 PRINT RND (5):
30 NEXT R

```

このプログラムを実行すると、0から5までの数字が20個並びます。一般に RND は RND (n) の形を取り、カッコの中の数字は、出したい乱数の最大値よりひとつ大きくします。ところで、サイコロは、1から6までの乱数ですね。ということは、

```
PRINT RND (5) + 1
```

↑  
0, 1, 2, 3, 4, 5の数のどれかが出ます。

にすればよいわけです。

さっそく“さいころ”のプログラムをつくってみましょう。

```

10 PRINT "さいころ"
20 PRINT RND (5) + 1:
30 IF INKEY$ <> "___" THEN 30
40 FOR W=0 TO 200:NEXT W
50 GOTO 20

```

スペースキーを押すたびに、数字がひとつ出ます。

note 1 / 20行で INKEY\$ を使っています。無限ループにしました

note 2 / 40行は、時間が過ぎるために入れた一行です。

INKEY\$ の関数はとてもはやいので、このようにしないと、一度にたくさんの数  
が次々に画面に出てしまいます。



RND 文の応用はたくさんあります。

- 座標を乱数できめて図形を描く  
LINE, CIRCLE, PSET 等
- キャラクターの動きをデタラメにする。
- サイコロやジョーンセンに使う。

など、ゲームに使うと意外性がでておもしろくなります。

RND 文を使ったグラフィックスのサンプルプログラムをあげておきます。参考にしてください。

#### サンプルプログラム ①

星々(きれいですよ)

```
10 SCREEN 2:CLS
20 FOR N=0 TO 200
30 X=RND(255)
40 Y=RND(191)
50 C=RND(16)
60 PSET(X,Y),C
70 NEXT N
```

座標と色を乱数で作ります。

#### サンプルプログラム ②

丸がいっぱい

```
10 SCREEN 2:CLS
20 FOR N=0 TO 30
30 X=RND(230)
40 Y=RND(170)
50 C=RND(16):IF C<2 THEN 60
60 R=RND(30)
70 CIRCLE(X,Y),R,C
80 NEXT N
```

座標と色と半径を乱数で作ります。

## SYSTEM (システム)

ページングからメニューにもとります。

シューティングゲームの背景をページングで作ったら、SYSTEM 文をプログラムの終わりに付け加えておけば自動的にメニューにもとります。

### 例

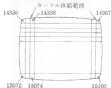
```
1000 REM ---- SYSTEM 424 ----
1100 SCREEN 2:CLS
1200 Y=0:YY=1/1
1300 FOR X=0 TO 255 STEP 1
1400 LINE X,YY-(X,YY),15
1500 NEXT X
1600 XX=0:XX=255
1700 FOR YY=0 TO 1 STEP 1
1800 LINE X,YY-(XX,YY),15
1900 NEXT Y
2000 SCREEN 1
2100 SYSTEM
```

## VPOKE ( ブイポーク )、VPEEK ( ブイピーク )

### \*\*\* VPOKE \*\*\*

テキスト画面には座標があることを CURSOR 文の項で説明しました。テキスト画面には座標の他にアドレス(番地)もあります。VPOKE 文ではアドレスを使います。指定したテキスト画面のアドレスに文字や記号を書き込むのです。

テキスト画面のアドレス



アドレスは、カーソルの移動できる範囲より左に 1、右に 1 広くなっています。

ブラウザ等の形状によっては文字が見えなくなることがあるので、カーソル移動の範囲をせめて表示領域を内側に寄せているのです。(例)

VPOKE 14336, 127

これでは、ほとんど見えません。

VPOKE 14333, 127

画面の左上に表示されます。

このように、

**VPOKE アドレス、キャラクタコード**

と入力すると、その場所に文字や記号を書き込むことができます。

# \*\*\* VPEEK \*\*\*

テキスト画面に何か書き込まれているか調べます

(例)

```
VPOKE 14880, 65
```

を実行すると画面に A が書き込まれます

```
PRINT VPEEK (14880)
```

を実行すると 65 と書き込まれます

キャラクターコード 65 を VPOKE で書き込むと A の文字になり、VPEEK で読むと 65 と数字で返ってきます。

VPEEK はテキスト画面でゲームを作る場合に、キャラクターを見分けることができるので便利です。

VPOKE と VPEEK のサンプルプログラム

```
10 CLS
20 FOR N=0 TO 50
30 V=RND(27)
40 W=RND(20)
50 C=RND(255)
60 IF C<=32 THEN 80
70 CURSOR V,W:PRINT CHR$(C)
80 NEXT N
```

画面にランダムに文字や記号を散らばめます。

```

90 CURSOR 0,20:PRINT " _____
_____ "
100 X=14338
110 VPOKE X,127
120 P=VPEEK(X+1)
130 CURSOR 16,21:PRINT CHR$(P)
140 FOR W=0 TO 30:NEXT W
150 VPOKE X,32
160 X=X+1:IF X=14872THEN END
170 IF P< > 32 THEN BEEP,GOTO200
180 GOTO 110
190 FOR W=0 TO 500:NEXT W
210 CURSOR16, 21:PRINT "
220 GOTO 110

```

VPOKEで書き込んだ ■  
 が移動していきます  
 文字や記号に ■ があつた  
 ると  
 とうなるでしう



■ がぶつかって電子や記号を渡します。

## ジョイスティックを使って

(ジョイスティックをお持ちの方)

STICK (n) と STRIG (n) はジョイスティックに関係のある命令文です。

### \*\*\* STRIG (スティックトリガ) \*\*\*

ジョイスティックのボタン・スイッチの状態を見ます。

使い方は

IF STRIG (n) =  THEN ~

STRIG (1)     1 Player 用のジョイスティック

STRIG (2)     2 Player 用のジョイスティック

 の所に入れる数値

|       |             |              |
|-------|-------------|--------------|
| 0 の場合 | OFF (オフ)    | 押されていません。    |
| 1 #   | 左 ON (オン)   | 左が押されています。   |
| 2 #   | 右 ON (オン)   | 右が押されています。   |
| 3 #   | 左、右 ON (オン) | 左、右が押されています。 |

### \*\*\* STICK (スティック) \*\*\*

ジョイスティックの状態を見ます。この命令でキャラクターやスプライトを操作しましょう。

使い方は

IF STICK (n) =  THEN ~

STICK (1) 1 player 用のジョイスティック

STICK (2) 2 player 用のジョイスティック

□□□ の所に入れる数値



□□□ の値を図の方向の値にすれば、その方向にレバーを倒したことになります。

たとえば

IF STICK (1) = 3 THEN ~  
(レバーを右に倒したら ~)

IF STICK (1) = 7 THEN ~  
(レバーを左に倒したら ~)

このように使います。

サンプルプログラムを参考にして下さい。



### サンプルプログラム

```
10 SCREEN 2:CLS
20 X=120:Y=80
30 PATTERN $#180,"183C7EFFFF7E3C18"
40 $1=STICK(1)
50 SPRITE 0,(X,Y),150,5
60 IF $1=1 THEN E=-1:GOTO 200
70 IF $1=3 THEN E=1:GOTO 300
80 IF $1=5 THEN E=1:GOTO 200
90 IF $1=7 THEN E=-1:GOTO 300
100 GOTO 40
200 Y=Y+E:IF Y<0 OR Y>180 THEN Y=Y-E
210 GOTO 100
300 X=X+E:IF X<0 OR X>255 THEN X=X-E
310 GOTO 100
```

これは基本型です。30行と40行の間や40行と50行の間にいろいろの変数を入れます。

```
25 V=100:W=30
35 PATTERN $#180,"FF00FF00FF00FF00"
55 SPRITE 1,(V,W),180,8
95 V=V+1
```

少し高度なことを知りたい人に

## CALL, POKE, LIMIT, PEEK

機械語という言葉があります。BASICは英語ですが、私たちの日常の言葉に近いのでわかりやすく初心者向きです。でも機械語は少しやっかいです。読書法ですし、コンピュータの仕組みがわからないと、なかなか使えないからです。

ここでは、機械語プログラムについては、あまり詳しく触れません。ただ、コンピュータ内部のメモリ（情報をたくわえておくところ）を分ければ、BASICプログラムだけでなく機械語プログラムもつくれるのだということを覚えておいてください。

### \*\*\* LIMIT (リミット) \*\*\*

BASIC プログラム領域の終了番地を設定します。

BASIC プログラム領域を設定します。LIMIT 文で設定した番地以降は BASIC インタプリタからの影響を受けずに使用可能になります。

*note /* 「番地」

メモリには番地がつけられています。これは、ちょうど住所の番地のような働きをします。つまり、メモリ内の特定の場所を指定するのです。

ページ内の行番号とは違います。

\*\*\* POKE (ポーク) \*\*\*

POKE **[番地]** , **[データ]**

の形で、指定したメモリの番地にデータを書き込みます。プログラム領域を十分に把握したうえで、空いている領域に書き込んでください。

シュートゲームの内容を変えるときに使えます。

POKE (&H830F) , &H1 **[CR]**

ペーシッタでこのように打ち込んでから、シュートゲームにもどってゲームをしてください。プレイヤーキャラクタが動く方向に弾が出るようになります。

\*\*\* CALL (コール) \*\*\*

BASIC プログラム領域外の、機械語プログラムを呼び出します。

一般に、

CALL **[機械番地]**

の形で入力します。

\*\*\* PEEK (ピーク) \*\*\*

番地で指定したメモリの内容を見せてくれます。

さきほど、POKE 文のところで &H830F の内容を変更しましたから、それを PEEK で確認してみましょう。

PRINT PEEK (&H830F) **[CR]**

**【例】** キーを押すと、1が表示されます。これは、

**P O K E & H 830 F , & H 1**

の1です。(10進法の1です。PEEK文では、結果は10進法で表示されるのです。)

PEEKは、上記のようにPRINT文で画面内容を見る他に、プログラムの中で使います。

## OUTとINP

データをコンピュータの外部へ出して、理解しやすい形に(プリントアウトする、或いは画面上に表示)することを出力(アウトプット)といいます。逆に、キーボードやデータレコードからデータを入れることを入力(インプット)といいます。データが出入りする場所をポートといいます。

### \*\*\* OUT (アウトポート) \*\*\*

出力ポートにデータを出力させます。一般に、

**OUT**   **【出力ポート番号】** ,   **【データ】**

の形をとります。 出力ポートには、プリント出力は何番、カセット出力は何番( ) というように、あらかじめ指定された番号がついています。(例)   **データレジスタ**   **&H DE**   **コマンドレジスタ**  
**&H BF**   **サケンドジュネレート**   **&H 7F**

出力ポート番号は0～255(&H 00～&H FF)までの整数です。

★★★ INP (インポート) ★★★

I/Oポート番号を指定して、ポート上にあるデータを読み出します。

(例)

```
10 A=INP(&HBE)
20 PRINT A
RUN
32
```

行番号 10で、I/Oポート番号 BE(16進数)にあるデータを、変数Aに読み出します。

*note* / I/Oポート番号は、システム中ですでに決定している番号で、0～255  
(&H00～&HFF)までの数値です。

## サンプルプログラム

### イロのテスト

```
10 REM "イロのテスト"
20 SCREEN 1:CLS
30 X=24:Y=16:R=12:G=2
40 CIRCLE(X,Y),R,"",100,0,100,
BF
50 X=X+30
55 C=C+1:IF C=16 THEN C=2
60 IF X=280 THEN X=0:Y=Y+16
70 IF Y=144 THEN GOTO 90
80 GOTO 40
90 GOTO 50
```

### イタズラがキ

```
10 REM "イタズラがキ"
20 SCREEN 2:CLS
30 RX=0:RY=0
40 C=0
50 IF C=0 OR C=1 THEN 40
60 LINE=(RX,RY),C
65 FOR N=0 TO 500:NEXT N
70 GOTO 30
```

## イロのボール

```

10  DIM A(10),C(10)
20  C=0: A=0: R=1: S=0
40  GOTO 100 IF R=10: GOTO 100 IF R=10:
50
60  A=R+C
70  C=A+10 IF C=10: R=R+1
80  IF A=100 THEN PRINT A: GOTO 100
90  IF C=10: R=R+1
100 GOTO 40

```

## エシをカウ

```

10 INPUT "I/P 年" Y
20 PRINT C=0:H=0:S=0:E=0
30 PRINT " シリヤ カノ"
40 INPUT "I/P 年" (R)=1: GOTO 100
50 INPUT "I/P 年" (C)=1: GOTO 100
60 INPUT "I/P 年" (H)=1: GOTO 100
70 INPUT "I/P 年" (S)=1: GOTO 100
80 INPUT "I/P 年" (E)=1: GOTO 100
90 PRINT " シリヤ カノ"
100 PRINT " シリヤ カノ"
110 PRINT " シリヤ カノ"
120 PRINT " シリヤ カノ"
130 PRINT " シリヤ カノ"
140 PRINT " シリヤ カノ"
150 PRINT " シリヤ カノ"
160 PRINT " シリヤ カノ"
170 PRINT " シリヤ カノ"
180 PRINT " シリヤ カノ"
190 PRINT " シリヤ カノ"
200 PRINT " シリヤ カノ"
210 PRINT " シリヤ カノ"
220 PRINT " シリヤ カノ"
230 PRINT " シリヤ カノ"
240 PRINT " シリヤ カノ"
250 PRINT " シリヤ カノ"
260 PRINT " シリヤ カノ"
270 PRINT " シリヤ カノ"
280 PRINT " シリヤ カノ"
290 PRINT " シリヤ カノ"
300 PRINT " シリヤ カノ"
310 PRINT " シリヤ カノ"
320 PRINT " シリヤ カノ"
330 PRINT " シリヤ カノ"
340 PRINT " シリヤ カノ"
350 PRINT " シリヤ カノ"
360 PRINT " シリヤ カノ"
370 PRINT " シリヤ カノ"
380 PRINT " シリヤ カノ"
390 PRINT " シリヤ カノ"
400 PRINT " シリヤ カノ"
410 PRINT " シリヤ カノ"
420 PRINT " シリヤ カノ"
430 PRINT " シリヤ カノ"
440 PRINT " シリヤ カノ"
450 PRINT " シリヤ カノ"
460 PRINT " シリヤ カノ"
470 PRINT " シリヤ カノ"
480 PRINT " シリヤ カノ"
490 PRINT " シリヤ カノ"
500 PRINT " シリヤ カノ"
510 PRINT " シリヤ カノ"
520 PRINT " シリヤ カノ"
530 PRINT " シリヤ カノ"
540 PRINT " シリヤ カノ"
550 PRINT " シリヤ カノ"
560 PRINT " シリヤ カノ"
570 PRINT " シリヤ カノ"
580 PRINT " シリヤ カノ"
590 PRINT " シリヤ カノ"
600 PRINT " シリヤ カノ"
610 PRINT " シリヤ カノ"
620 PRINT " シリヤ カノ"
630 PRINT " シリヤ カノ"
640 PRINT " シリヤ カノ"
650 PRINT " シリヤ カノ"
660 PRINT " シリヤ カノ"
670 PRINT " シリヤ カノ"
680 PRINT " シリヤ カノ"
690 PRINT " シリヤ カノ"
700 PRINT " シリヤ カノ"
710 PRINT " シリヤ カノ"
720 PRINT " シリヤ カノ"
730 PRINT " シリヤ カノ"
740 PRINT " シリヤ カノ"
750 PRINT " シリヤ カノ"
760 PRINT " シリヤ カノ"
770 PRINT " シリヤ カノ"
780 PRINT " シリヤ カノ"
790 PRINT " シリヤ カノ"
800 PRINT " シリヤ カノ"
810 PRINT " シリヤ カノ"
820 PRINT " シリヤ カノ"
830 PRINT " シリヤ カノ"
840 PRINT " シリヤ カノ"
850 PRINT " シリヤ カノ"
860 PRINT " シリヤ カノ"
870 PRINT " シリヤ カノ"
880 PRINT " シリヤ カノ"
890 PRINT " シリヤ カノ"
900 PRINT " シリヤ カノ"
910 PRINT " シリヤ カノ"
920 PRINT " シリヤ カノ"
930 PRINT " シリヤ カノ"
940 PRINT " シリヤ カノ"
950 PRINT " シリヤ カノ"
960 PRINT " シリヤ カノ"
970 PRINT " シリヤ カノ"
980 PRINT " シリヤ カノ"
990 PRINT " シリヤ カノ"
1000 PRINT " シリヤ カノ"

```





|     |     |
|-----|-----|
| 100 | 100 |
| 101 | 101 |
| 102 | 102 |
| 103 | 103 |
| 104 | 104 |
| 105 | 105 |
| 106 | 106 |
| 107 | 107 |
| 108 | 108 |
| 109 | 109 |
| 110 | 110 |
| 111 | 111 |
| 112 | 112 |
| 113 | 113 |
| 114 | 114 |
| 115 | 115 |
| 116 | 116 |
| 117 | 117 |
| 118 | 118 |
| 119 | 119 |
| 120 | 120 |
| 121 | 121 |
| 122 | 122 |
| 123 | 123 |
| 124 | 124 |
| 125 | 125 |
| 126 | 126 |
| 127 | 127 |
| 128 | 128 |
| 129 | 129 |
| 130 | 130 |
| 131 | 131 |
| 132 | 132 |
| 133 | 133 |
| 134 | 134 |
| 135 | 135 |
| 136 | 136 |
| 137 | 137 |
| 138 | 138 |
| 139 | 139 |
| 140 | 140 |
| 141 | 141 |
| 142 | 142 |
| 143 | 143 |
| 144 | 144 |
| 145 | 145 |
| 146 | 146 |
| 147 | 147 |
| 148 | 148 |
| 149 | 149 |
| 150 | 150 |
| 151 | 151 |
| 152 | 152 |
| 153 | 153 |
| 154 | 154 |
| 155 | 155 |
| 156 | 156 |
| 157 | 157 |
| 158 | 158 |
| 159 | 159 |
| 160 | 160 |
| 161 | 161 |
| 162 | 162 |
| 163 | 163 |
| 164 | 164 |
| 165 | 165 |
| 166 | 166 |
| 167 | 167 |
| 168 | 168 |
| 169 | 169 |
| 170 | 170 |
| 171 | 171 |
| 172 | 172 |
| 173 | 173 |
| 174 | 174 |
| 175 | 175 |
| 176 | 176 |
| 177 | 177 |
| 178 | 178 |
| 179 | 179 |
| 180 | 180 |
| 181 | 181 |
| 182 | 182 |
| 183 | 183 |
| 184 | 184 |
| 185 | 185 |
| 186 | 186 |
| 187 | 187 |
| 188 | 188 |
| 189 | 189 |
| 190 | 190 |
| 191 | 191 |
| 192 | 192 |
| 193 | 193 |
| 194 | 194 |
| 195 | 195 |
| 196 | 196 |
| 197 | 197 |
| 198 | 198 |
| 199 | 199 |
| 200 | 200 |



## CURSOR コントロール

```

5 REM *-- コールバック
10 CLS
20 X=15:Y=10:F=1
100 CURSOR X,Y:PRINT "A"
105 Z#=INKEY#
110 IF Z#=CHR$(28) THEN F=1:GOSUB 200
120 IF Z#=CHR$(29) THEN F=-1:GOSUB 200
130 IF Z#=CHR$(30) THEN F=-1:GOSUB 300
140 IF Z#=CHR$(31) THEN F=1:GOSUB 300
160 CURSOR X,Y:PRINT "A"
170 GOTO 100
'000 RPR
'005 CURSOR X,Y:PRINT " "
210 X=X+F:IF X=0 OR X=29 THEN X=X-F
'020 RETURN
300 REM
'025 CURSOR X,Y:PRINT " "
'10 Y=Y+F:IF Y=0 OR Y=23 THEN Y=Y-F
'10 RETURN

```

# VPOKE コントロール

```

5 REM ← 340-6
10 GOTO 100
20 V=147:G:A=65:F=11:T=16:S=12
120 VPOKE V,A
105 Z#=INKEY$
110 IF Z#=CHR$(28) THEN E=1:GOSUB 200
120 IF Z#=CHR$(29) THEN E=-1:GOSUB 200
130 IF Z#=CHR$(30) THEN F=32:GOSUB 300
140 IF Z#=CHR$(31) THEN F=32:GOSUB 300
170 GOTO 100
200 REM
205 VFO=V,V,32
210 V=V+E:T=T+E:IF T=2 OR T=29 THEN V=
V-E
220 RETURN
300 REM
305 VFO=V,V,32
310 V=V+F
320 RETURN

```

## スプライトのしょうとつ

```

100 SCREEN=0,CLS
110 X=0,Y=0: X2=240:Y2=90
200 F=0:GOTO S#0, FF02FF02FF02FF02
F02
300 PATTERN S#4,"AAAA000000AA-
AA"
400 SPRING=0,X=0,Y=0,5
510 SPRINGL=4,(X,Y),4,8
1=0 X=X+1
510 IF SPRING=-1 THEN GOSUB 20
0
190 GOTO 100
200 REM
210 X=0
220 P=0:DEF:BE:CF
230 GOTO 100

```

## スプライト テスト

```

5 REM *****
10 SCREEN 2:CLS
20 X=40:Y=90:X=X+170:Y=Y-90
30 M=1
50 PATTERN S#0,"00000000000000
000"
51 PATTERN S#1,"00000000000000
000"
52 PATTERN S#2,"FFFFFF0707070
707"
53 PATTERN S#3,"070707070707FF
FFF"
54 PATTERN S#4,"00000000000000
0FF"
55 PATTERN S#5,"FFFF0000000000
000"
56 PATTERN S#6,"0000000010101
0FE"
57 PATTERN S#7,"FFFE1C1010000
000"
70 PATTERN S#8,"FFF=FFE0E0E0E
0E0"

```

```

1 PATTERN S#9,"E0E0E0E0E0FF=
0E0"
2 PATTERN S#10,"000000000000
0000"
3 PATTERN S#11,"000000000000
0000"
100 MAG M
110 SPRITE 0,(X,Y),0,3
120 SPRITE 2,(X,Y),0,3
130 SPRITE 1,(X,Y),4,0
140 X=X+1:IF X=255 THEN GOSUB
200
190 GOTO 110
200 X=0
210 M=M+2:IF M=3 THEN M=1
220 MAG M
230 RETURN

```

# ニゲロニゲロ

```

10 REM --- 1.1.1 ---
20 SCREEN 2:CLS
30 GOTO 1000
40 W=120:H=90
50 V=1:W=90
60 DEF FN F1(X,Y) = (24*(191+Y),Y)
70 MAG 0
80 REM --- 1.1.2 ---
90 Z#=INI EY#
100 IF Z#=CHR$(28) THEN F=2160
110 GOTO 700
120 IF Z#=CHR$(29) THEN F=216
130 SUB 300
140 IF Z#=CHR$(70) THEN F=216
150 SUB 400
160 IF Z#=CHR$(71) THEN F=2160
170 SUB 400
180 REM --- 1.1.3 ---
190 SPRITE 0,(X,Y),200,0
200 SPRITE 16,V,W,216,0
210 IF V=X THEN E=-1
220 IF Y THEN E=1
230 IF W=Y THEN T=1
240 IF W=Y THEN D=-1

```

```

250 V=V+E+W-W-E
260 REM --- 1.1.4 ---
270 IF SOUND THEN BE PIGOT 6
280
290 GOTO 100
300 REM --- 1.1.5 ---
310 Z=X+F:IF X 16 OR X 272 TH
EN Z=X-F
320 RETURN
330 REM --- 1.1.6 ---
340 Y=Y+F:IF Y 0 OR Y 175 THE
N Y=Y-F
350 RETURN
360 REM --- 1.1.7 ---
370 FOR R=5 TO 40 STEP 5
380 CIRCLE(X,Y),R,10
390 CIRCLE(X,Y),R,17
400 NEXT R
410 GOTO 70
420 PATTERNS#000,"700010FE10
10264"
430 PATTERNS#016,"19CCAD0BFF
244291"
440 RETURN

```





```

510  CALL
520  C1=C1+5*(12*(4*(1/1000
530  )-32*(1/1000)+12*(4*(1/100
540  )-12*(1/1000)
550  4*(1/1000)+1*(1/1000)+4*(1/100
560  )-12*(1/1000)+1*(1/1000)+3*(1/1
570  )-12*(1/1000)
580  C1=C1+5*(12*(4*(1/1000)+1*(1/1000)+4*(1/1000)
590  24*(1/1000)
600  IF 740=1 THEN GOTO 610
610  GOTO 1000
620  REM ---
630  4*(1/1000)+1*(1/1000)+4*(1/1000)
640  4*(1/1000)+1*(1/1000)+4*(1/1000)
650  4*(1/1000)+1*(1/1000)+4*(1/1000)
660  4*(1/1000)+1*(1/1000)+4*(1/1000)
670  4*(1/1000)+1*(1/1000)+4*(1/1000)
680  4*(1/1000)+1*(1/1000)+4*(1/1000)
690  4*(1/1000)+1*(1/1000)+4*(1/1000)
700  4*(1/1000)+1*(1/1000)+4*(1/1000)
710  4*(1/1000)+1*(1/1000)+4*(1/1000)
720  4*(1/1000)+1*(1/1000)+4*(1/1000)
730  4*(1/1000)+1*(1/1000)+4*(1/1000)
740  4*(1/1000)+1*(1/1000)+4*(1/1000)
750  4*(1/1000)+1*(1/1000)+4*(1/1000)
760  4*(1/1000)+1*(1/1000)+4*(1/1000)
770  4*(1/1000)+1*(1/1000)+4*(1/1000)
780  4*(1/1000)+1*(1/1000)+4*(1/1000)
790  4*(1/1000)+1*(1/1000)+4*(1/1000)
800  4*(1/1000)+1*(1/1000)+4*(1/1000)
810  4*(1/1000)+1*(1/1000)+4*(1/1000)
820  4*(1/1000)+1*(1/1000)+4*(1/1000)
830  4*(1/1000)+1*(1/1000)+4*(1/1000)
840  4*(1/1000)+1*(1/1000)+4*(1/1000)
850  4*(1/1000)+1*(1/1000)+4*(1/1000)
860  4*(1/1000)+1*(1/1000)+4*(1/1000)
870  4*(1/1000)+1*(1/1000)+4*(1/1000)
880  4*(1/1000)+1*(1/1000)+4*(1/1000)
890  4*(1/1000)+1*(1/1000)+4*(1/1000)
900  4*(1/1000)+1*(1/1000)+4*(1/1000)
910  4*(1/1000)+1*(1/1000)+4*(1/1000)
920  4*(1/1000)+1*(1/1000)+4*(1/1000)
930  4*(1/1000)+1*(1/1000)+4*(1/1000)
940  4*(1/1000)+1*(1/1000)+4*(1/1000)
950  4*(1/1000)+1*(1/1000)+4*(1/1000)
960  4*(1/1000)+1*(1/1000)+4*(1/1000)
970  4*(1/1000)+1*(1/1000)+4*(1/1000)
980  4*(1/1000)+1*(1/1000)+4*(1/1000)
990  4*(1/1000)+1*(1/1000)+4*(1/1000)
1000 4*(1/1000)+1*(1/1000)+4*(1/1000)

```

# マシンとボム

```

100 IF P = 0 THEN GOTO 100
110 GOTO 100
120 GOTO 100
130 LINE(10,0)-(100,100),15,0
140 LINE(10,100)-(100,100),15,0
150 LINE(10,100)-(100,100),15,0
160 PRINT:GOTO 100
170 REM
180 PATTERN 5#200,"870F340000C
190 PATTERN 5#201,"1000000000
200 PATTERN 5#202,"1000000000
210 PATTERN 5#203,"1000000000
220 PATTERN 5#204,"1000000000
230 PATTERN 5#205,"1000000000
240 PATTERN 5#206,"1000000000
250 PATTERN 5#207,"1000000000

```

```

160 PATTERN 5#207,"1000000000
170 PATTERN 5#208,"1000000000
180 PATTERN 5#209,"1000000000
190 PATTERN 5#210,"1000000000
200 PATTERN 5#211,"1000000000
210 REM
220 COLOR 1,3
230 PRINTCHR$(17)
240 PRINT:PRINT      S E G A
250 PRINT:PRINT      HOME BASIC
260 X=100:Y=100:F=100:G=100:H=100
270 HNS H
280 SHLLE 10,(X,Y),P,4
290 SPITE 1,(X,Y),204,15

```

```

400 F=F+1:Y=Y+60:F1=140:B
710 F1=B:IF F1=0 THEN F=200
720 F=F+7:IF F=100 THEN F=1
570 IF F=91 THEN F=-1
410 X=X+1:IF X=8 THEN X=255:
8010 420
150 GOTO
360 GOTO 270
70 PATTERN 5#160,"050F3F3F7F
7FF0FF"
310 PATTERN 0#161,"FFFF7F7F3F
0F0F0F"
190 PATTERN 6#162,"00F01CCE6
F4F0FB"
400 PATTERN 8#163,"FF0F0F0F0
LCF0C0"
410 RETURN
400 M=M+2:IF M=3 THEN M=1
50 GOTO 270

```



```

5150 PATTERN 54191,"00F901000"
5160 PATTERN 54200,"070000010"
5170 PATTERN 54001,"013F7F7F7"
5180 PATTERN 54202,"F0E282C0C"
5190 PATTERN 54203,"E2FEFFFFF"
5200 RETURN
5210 REM --- 5270- 28 5150 ---
5210 S1=200:52=100
5220 SFX(1)=1,X(1)=0,Y(1)=1,5
5230 SFX(2)=4,X(2)=1,Y(2)=0
5240 Y=Y+1:Y=Y+1:IF Y=180 TH
FN 5210
5250 SOUND 4,2,15
5260 FOR H=0 TO 1000:NEX H
5270 X=200
5280 H1=0:H=0:51=180:52=100
5290 GOTO 50
5300 GOTO 10
5310 REM --- 28 5210 ---
5320 VY=V+0.01:VY=INT(VY*2
5330 IF VY<4 THEN H=0
5340 LINE(X,VY)=(H,VY),15
5350 IF VY=4 OR VY=180 THE
N GOTO 5320
5360 IF X=H1 THEN 5330
5370 LINE(X,VY)=(H1,VY),0.01
5380 GOTO 110
5390 REM --- 28 5210 ---
5400 S1=200
5410 LINE(X,Y)=(H1,VY),0.01
5420 FOR H=20 TO 1000 S1=S1+
5430 SFX(2),X(Y),0,0
5440 SOUND 1,H,15
5450 Y=Y+1:IF Y=180 GOTO 5400
5460 NEXT H
5470 H1=0:X=0:51=100
5480 SOUND 0:GOTO 10

```

# 変数・配列

|   |       |    |                                         |
|---|-------|----|-----------------------------------------|
| { | 数値変数  | —— | A, B, ~ Z, AA, AB, ~ ZZ<br>AO, A1 ~ A9  |
|   | 数値配列  | …… | 2次元まで A(15), B(5, 5),<br>AC(3, 3)       |
|   | 文字列変数 | …… | A\$, AB\$, A1\$                         |
|   | 文字列配列 | …… | 2次元まで A\$(15),<br>B\$(5, 5), AC\$(3, 3) |

- 変数名は、先頭の1文字を英字、以降を英字または数字とする。長さは何文字でも良いが先頭の2文字で区別する。
- 変数と配列の名前が同じでも良い。

## <文字列変数、配列の範囲>

文字の長さ      0 → 255

## 定 数

### ＜数値定数＞

- ・ 整数形式 (例) 3, -2, 9992
- ・ 16進形式 &H 16進の数 ..... 0000 ~ FFFF  
(例) &H 04 ..... 16進と同じ  
&H FFFF ..... -1と同じ  
数値の範囲 + 32767 ~ - 32768  
&H 8000 ~ &H 7FFF
- ・ 負の最大値の表記について  
負の最大値 (-32768) はそのまま表記するとオーバーフローエラーとなる。  
(理由: -32768は定数 32768にマイナスの単項演算子を行ったものと)  
解釈されるため。  
負の最大値は次の様に表記する。  
(-1-32767) 又は &H 8000

### ＜文字列定数＞

- ・ で囲む。 は を2つ囲んで示す。  
(例) "ABC" → 文字ABC  
" " → 文字空白LL(なし)  
" " → 文字"  
"A3" "64" → 文字A3"64

| 内 容                     | 対 数    |
|-------------------------|--------|
| 両面から内部に取り込まれる文字数        | 255 文字 |
| 文字列として扱うことの可能な文字数       | 31 文字  |
| FOR ~ NEXT のネストレベルの数    | 8 レベル  |
| GOSUB, RETURN のネストレベルの数 | 16 レベル |



## **CTRL** (コントロール)

**CTRL** キーを押したまま、文字キーを打ちますと、表にある動作が行われます。

### コントロールコード

| キー操作                   | PRINT CHR\$( 値 ); | 機 能                      |
|------------------------|-------------------|--------------------------|
| <b>CTRL</b> + <b>A</b> | PRINT CHR\$( 1 ); | NULL 文字なし                |
| <b>C</b>               | —                 | BREAK プログラムの実行中止         |
| <b>E</b>               | 5                 | カーソル以降の文字をクリア            |
| <b>G</b>               | 7                 | BELL ビープ音を出す             |
| <b>H</b>               | 8                 | DEL 文字をデリート              |
| <b>I</b>               | 9                 | HT 水平 TAB                |
| <b>J</b>               | 10                | LF ラインフェード               |
| <b>K</b>               | 11                | HM カーソルを左上端に戻す HOME      |
| <b>L</b>               | 12                | CLR 画面のクリア HOME          |
| <b>M</b>               | 13                | CR キャリッジリターン             |
| <b>N</b>               | 14                | カナ == 英数字切りかえ            |
| <b>O</b>               | 15                | /? 画面をテキスト == グラフィック切りかえ |

| キー操作 | PRINT CHR\$(値) | 機 能                     |
|------|----------------|-------------------------|
| P    | 16             | 標準文字サイズ                 |
| Q    | 17             | 縮小文字サイズ (SCREEN 2)      |
| R    | 18             | INS インスर्ट              |
| S    | 19             | キー入力(A~Z)ノリなし大文字        |
| T    | 20             | キー入力(a~z)ノリなし小文字        |
| U    | 21             | ラインをクリアしカーソルを左端に戻す。     |
| V    | 22             | ノーマルモードすべて初期の状態にする。     |
| W    | 23             | GRAPH キー入力グラフモード ↔ 英字切換 |
| X    | 24             | グラフィックの ON ↔ OFF 切換     |
| —    | 28             | ⇒ カーソル移動                |
| —    | 29             | ←                       |
| —    | 30             | ↑                       |
| —    | 31             | ↓                       |

プログラム中でコントロールコードを使う場合は、

PRINT CHR\$(値)で入力して下さい。

キャラクターコード

|   | 0 | 1 | 2  | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 |   |   |    | 0 | @ | P | + | p | + | + | + | + | + | + | + | + |
| 1 |   |   | !  | 1 | A | Q | a | q | + | あ | 。 | ア | チ | ム | ち | む |
| 2 |   |   | "  | 2 | B | R | b | r | + | い | 「 | イ | ツ | メ | つ | め |
| 3 |   |   | #  | 3 | C | S | c | s | + | う | 」 | ウ | テ | モ | て | も |
| 4 |   |   | \$ | 4 | D | T | d | t | + | え | 、 | エ | ト | ヤ | と | や |
| 5 |   |   | %  | 5 | E | U | e | u | + | お | ・ | オ | ナ | ユ | な | ゆ |
| 6 |   |   | &  | 6 | F | V | f | v | + | を | か | ワ | カ | ニ | こ | よ |
| 7 |   |   | '  | 7 | G | W | g | w | + | き | ア | キ | ヌ | ラ | ぬ | ら |
| 8 |   |   | (  | 8 | H | X | h | x | + | く | イ | ク | ホ | リ | こ | り |
| 9 |   |   | )  | 9 | I | Y | i | y | + | け | ウ | ケ | ノ | ル | の | る |
| A |   |   | *  | A | Z | J | z | j | + | こ | エ | コ | ハ | レ | は | れ |
| B |   |   | +  | B | [ | k | [ | k | + | ま | ア | サ | ビ | ロ | ひ | ろ |
| C |   |   | ,  | C | L | N | l | n | + | し | サ | シ | フ | ワ | ふ | わ |
| D |   |   | -  | D | M | O | m | o | + | す | ホ | ス | ヘ | ン | へ | ん |
| E |   |   | .  | E | N | ハ | n | h | + | せ | サ | セ | ホ | 。 | ほ | 。 |
| F |   |   | /  | F | O | カ | o | + | っ | そ | ノ | ソ | サ | 。 | ま | 。 |

コード = 4 \* コード

# キ + ラクナーコード

|    |    |    |   |    |   |    |   |     |   |     |   |     |   |     |   |
|----|----|----|---|----|---|----|---|-----|---|-----|---|-----|---|-----|---|
| 32 | SP | 46 | 0 | 64 | a | 80 | P | 96  |   | 112 | p | 128 |   | 144 | í |
| 33 | !  | 49 | 1 | 65 | A | 81 | Q | 97  | q | 113 | q | 129 |   | 145 | あ |
| 34 | •  | 50 | 2 | 66 | B | 82 | R | 98  | b | 114 | r | 130 |   | 146 | い |
| 35 | \$ | 51 | 3 | 67 | C | 83 | S | 99  | s | 115 | s | 131 |   | 147 | う |
| 36 | \$ | 52 | 4 | 68 | D | 84 | T | 100 | d | 116 | t | 132 |   | 148 | え |
| 37 | %  | 53 | 5 | 69 | E | 85 | U | 101 | e | 117 | u | 133 |   | 149 | お |
| 38 | &  | 54 | 6 | 70 | F | 86 | V | 102 | f | 118 | v | 134 | を | 150 | か |
| 39 | ¥  | 55 | 7 | 71 | G | 87 | W | 103 | g | 119 | w | 135 | あ | 151 | き |
| 40 | !  | 56 | 8 | 72 | H | 88 | X | 104 | h | 120 | x | 136 | い | 152 | く |
| 41 | !  | 57 | 9 | 73 | I | 89 | Y | 105 | i | 121 | y | 137 | う | 153 | け |
| 42 | +  | 58 | : | 74 | J | 90 | Z | 106 | j | 122 | z | 138 | え | 154 | こ |
| 43 | +  | 59 | ; | 75 | K | 91 | [ | 107 | k | 123 | ! | 139 | あ | 155 | さ |
| 44 | .  | 60 | , | 76 | L | 92 | * | 108 | ! | 124 | ! | 140 | や | 156 | し |
| 45 | -  | 61 | = | 77 | M | 93 | ] | 109 | m | 125 | ! | 141 | ゆ | 157 | ず |
| 46 | .  | 62 | > | 78 | N | 94 | ^ | 110 | n | 126 | ! | 142 | よ | 158 | せ |
| 47 | /  | 63 | ? | 79 | O | 95 | ~ | 111 | o | 127 | ! | 143 | つ | 159 | そ |

|     |   |     |   |     |   |     |   |     |   |     |   |
|-----|---|-----|---|-----|---|-----|---|-----|---|-----|---|
| 160 | □ | 176 | ニ | 192 | ク | 208 | ミ | 224 | た | 240 | み |
| 161 | □ | 177 | ア | 193 | チ | 209 | ム | 225 | ち | 241 | む |
| 162 | □ | 178 | イ | 194 | ツ | 210 | メ | 226 | つ | 242 | め |
| 163 | □ | 179 | ウ | 195 | ヂ | 211 | モ | 227 | て | 243 | も |
| 164 | □ | 180 | エ | 196 | ト | 212 | ヤ | 228 | と | 244 | や |
| 165 | □ | 181 | オ | 197 | ナ | 213 | ユ | 229 | な | 245 | ゆ |
| 166 | □ | 182 | カ | 198 | ニ | 214 | ヨ | 230 | に | 246 | よ |
| 167 | □ | 183 | キ | 199 | ス | 215 | ラ | 231 | ぬ | 247 | ら |
| 168 | □ | 184 | ク | 200 | ホ | 216 | リ | 232 | ね | 248 | り |
| 169 | □ | 185 | ケ | 201 | ノ | 217 | ル | 233 | の | 249 | る |
| 170 | □ | 186 | コ | 202 | ハ | 218 | レ | 234 | は | 250 | れ |
| 171 | □ | 187 | サ | 203 | ヒ | 219 | ロ | 235 | ひ | 251 | ろ |
| 172 | □ | 188 | シ | 204 | フ | 220 | ワ | 236 | ふ | 252 | わ |
| 173 | □ | 189 | ス | 205 | ヘ | 221 | ン | 237 | へ | 253 | ん |
| 174 | □ | 190 | セ | 206 | ホ | 222 | □ | 238 | ほ | 254 | □ |
| 175 | □ | 191 | ソ | 207 | マ | 223 | □ | 239 | ま | 255 | □ |

# BASICのSF PATTERNとパターンゼンゴとの関係

| BASIC | パターン | BASIC | パターン | BASIC | パターン | BASIC | パターン | BASIC | パターン |
|-------|------|-------|------|-------|------|-------|------|-------|------|
| 0     | 0    | 22    |      | 44    | 11   | 66    |      | 88    | 22   |
| 1     |      | 23    |      | 45    |      | 67    |      | 89    |      |
| 2     |      | 21    | 6    | 46    |      | 68    | 17   | 90    |      |
| 3     |      | 25    |      | 47    |      | 69    |      | 91    |      |
| 4     | 1    | 26    |      | 48    | 12   | 70    |      | 92    | 23   |
| 5     |      | 27    |      | 49    |      | 71    |      | 93    |      |
| 6     |      | 28    | 7    | 50    |      | 72    | 18   | 94    |      |
| 7     |      | 29    |      | 51    |      | 73    |      | 95    |      |
| 8     | 2    | 30    |      | 52    | 13   | 74    |      | 96    | 24   |
| 9     |      | 31    |      | 53    |      | 75    |      | 97    |      |
| 10    |      | 32    | 8    | 54    |      | 76    | 19   | 98    |      |
| 11    |      | 33    |      | 55    |      | 77    |      | 99    |      |
| 12    | 3    | 34    |      | 56    | 14   | 78    |      | 100   | 25   |
| 13    |      | 35    |      | 57    |      | 79    |      | 101   |      |
| 14    |      | 36    | 9    | 58    |      | 80    | 20   | 102   |      |
| 15    |      | 37    |      | 59    |      | 81    |      | 103   |      |
| 16    | 4    | 38    |      | 60    | 15   | 82    |      | 104   | 26   |
| 17    |      | 39    |      | 61    |      | 83    |      | 105   |      |
| 18    |      | 40    | 10   | 62    |      | 84    | 21   | 106   |      |
| 19    |      | 41    |      | 63    |      | 85    |      | 107   |      |
| 20    | 5    | 42    |      | 64    | 16   | 86    |      | 108   | 27   |
| 21    |      | 43    |      | 65    |      | 87    |      | 109   |      |

| BASIC | バシク | BASIC | バシク | BASIC | バシク | BASIC | バシク | BASIC | バシク | BASIC | バシク |
|-------|-----|-------|-----|-------|-----|-------|-----|-------|-----|-------|-----|
| 132   | 32  | 155   |     | 178   |     | 201   |     | 224   | 36  | 247   |     |
| 133   |     | 156   | 39  | 179   |     | 202   |     | 225   |     | 248   | —   |
| 134   |     | 157   |     | 180   | 45  | 203   |     | 226   |     | 249   |     |
| 135   |     | 158   |     | 181   |     | 204   | 51  | 227   |     | 250   |     |
| 136   | 34  | 159   |     | 182   |     | 205   |     | 228   | 57  | 251   |     |
| 137   |     | 160   | 40  | 183   |     | 206   |     | 229   |     | 252   | —   |
| 138   |     | 161   |     | 184   | 46  | 207   |     | 230   |     | 253   |     |
| 139   |     | 162   |     | 185   |     | 208   | 52  | 231   |     | 254   |     |
| 140   | 35  | 163   |     | 186   |     | 209   |     | 232   | 58  | 255   |     |
| 141   |     | 164   | 41  | 187   |     | 210   |     | 233   |     |       |     |
| 142   |     | 165   |     | 188   | 47  | 211   |     | 234   |     |       |     |
| 143   |     | 166   |     | 189   |     | 212   | 53  | 235   |     |       |     |
| 144   | 36  | 167   |     | 190   |     | 213   |     | 236   | 59  |       |     |
| 145   |     | 168   | 42  | 191   |     | 214   |     | 237   |     |       |     |
| 146   |     | 169   |     | 192   | 48  | 215   |     | 238   |     |       |     |
| 147   |     | 170   |     | 193   |     | 216   | 54  | 239   |     |       |     |
| 148   | 37  | 171   |     | 194   |     | 217   |     | 240   | —   |       |     |
| 149   |     | 172   | 43  | 195   |     | 218   |     | 241   |     |       |     |
| 150   |     | 173   |     | 196   | 49  | 219   |     | 242   |     |       |     |
| 151   |     | 174   |     | 197   |     | 220   | 55  | 243   |     |       |     |
| 152   | 38  | 175   |     | 198   |     | 221   |     | 244   | —   |       |     |
| 153   |     | 176   | 44  | 199   |     | 222   |     | 245   |     |       |     |
| 154   |     | 177   |     | 200   | 50  | 223   |     | 246   |     |       |     |

# メインメモリーマップ







## エラーメッセージ

### 1 表示方法

(1) シェット入力又はスクリプトをダイアログボックスで実行する際にエラーが発生した場合

? **メッセージ** error

(2) ツールストリッププログラムを実行中にエラーが発生した場合

? **メッセージ** error in **ラインナンバー**

(3) INPUT ステートメントによるキー入力データに誤りがあった場合

? **メッセージ**

## エラーメッセージ (アルファベット順)

|                       |                                             |
|-----------------------|---------------------------------------------|
| ARRAY NAME            | DIM 文のパラメータが配列でない                           |
| CAN'T CONTINUE        | CONT による続行が不可能                              |
| DEVICE NOT READY      | プリンタが接続されていない、もしくは、プリンタが故障している。             |
| DIVISION BY ZERO      | わり算の分母が0である。                                |
| EXTRA IGNORED         | INPUT 文の入力データがおかしい。                         |
| FOR NESTING           | FOR ~ NEXT の入れ子が8回を超えた。                     |
| FOR VARIABLE NAME     | FOR 文のあとの変数か数値変数でない。(文字などになっている。)適切な変数に変えよ。 |
| GOSUB NESTING         | GOSUB の入れ子が8回を超えた。                          |
| ILLEGAL DIRECT        | ダイレクト命令実行不可能                                |
| ILLEGAL FUNCTION CALL | 関数のパラメタ、命令文のパラメタがおかしい。                      |
| ILLEGAL LINE NUMBER   | 行番号がおかしい。                                   |
| LINE IMAGE TOO LONG   | ラインが長くなりすぎる。                                |
| MEMORY WRITING        | メモリへの書き込みエラー。                               |
| NEXT WITHOUT FOR      | NEXT に対応する FOR 文がない。                        |
| NO PROGRAM            | プログラムがないのに、SAVEしようとした。                      |
| NO VRAM-DATA          | VRAM データがないのに、SAVEしようとした。                   |

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| OUT OF DATA          | READ 定で読み終 (END OF DATA) であるのに                      |
| OUT OF MEMORY        | メモリ不足 (オーバーフロー)                                     |
| OVERFLOW             | 値や保存結果が許容範囲を越えた                                     |
| REDIM'D ARRAY        | 配列を途中に記憶しようとした                                      |
| READ FROM START      | INPUT 文の入力がおかしい (行 230 のように) して、                    |
| RETURN WITHOUT GOSUB | GOSUB を行わないで RETURN 文を実行された                         |
| STACK OVERFLOW       | ・カッコの使いすぎ 数値式・変数すぎる 　・文字列が複雑すぎる 　・PAINT する図形が複雑すぎる  |
| STRING TOO LONG      | 文字列が長すぎる (255 を越えた)                                 |
| SUBSCRIPT            | 添字の数 (個数)、或いは添字の値がおかしい                              |
| SYNTAX               | 文法 (構文) 上のエラー                                       |
| SYSTEM               | BASIC インタプリタのプログラムの動作エラー                            |
| TYPE MISMATCH        | 代入する側と代入される側のタイプが合わない (数値、文字列)                      |
| UNDEF'0 LINE NUMBER  | (GOTO, GOSUB, IF~THEN, RESTORE, RUN など) 指定された行番号がない |
| UNDEFINED ARRAY      | 未定義の配列を ERASE しようとした                                |
| UNPRINTABLE          | 上記以外のエラー                                            |

# コマンド・ステートメント・関数 索引

(アルファベット順)

|           |                                        |     |
|-----------|----------------------------------------|-----|
| ABS(X)    | Xの絶対値を求める。                             | 95  |
| ASC(S)    | 文字列Sの最初の文字のコードを数値で与える。                 | 95  |
| AUTO      | 行番号を自動的につける。                           | 60  |
| BEEP      | ビーブ音を発生させる。                            | 119 |
| CALL      | 機械語プログラムを呼び出す                          | 109 |
| CHR \$(X) | Xに対応する文字や機能を与える。                       | 92  |
| CIRCLE    | 円を描く。                                  | 127 |
| CLS       | プログラムは消さずに、画面を消す。                      | 50  |
| COLOR     | 色を設定する。                                | 131 |
| CONSOLE   | 画面スクロールの範囲を設定する。また、クリップ音の有無、英字の大小を求める。 | 65  |
| CONT      | 中断されていたプログラムを続行する                      | 50  |
| CURSOR    | 表示位置を設定する                              | 61  |
| DATA      | "READ"により読め込まれるデータを示す                  | 79  |
| DIM       | 配列を宣言する。                               | 85  |
| END       | プログラムの終了を示す。                           | 47  |
| ERASE     | 宣言された配列をクリアする。                         | 69  |

|                      |                                        |     |
|----------------------|----------------------------------------|-----|
| FOR ~ TO NEXT ~ STEP | 指定された条件で仕事をくりかえす。                      | 43  |
| FRE                  | とれてくさいメモリが残っていることを示す。                  | 63  |
| GO\$UB               | サブルーチンへ分岐する。                           | 76  |
| GOTO                 | 指定された行番号へとぶ。                           | 44  |
| HEX \$(X)            | 16進数文字列を与える。                           | 96  |
| IF ~ THEN            | 条件にしたがって仕事をくりかえす。(条件分岐)                | 72  |
| INKEY \$             | キーが押されたかどうか調べる。                        | 156 |
| INP                  | 入力ポートの入力内容を与える。                        | 171 |
| INPUT                | キーから入力するように要求する。                       | 68  |
| LEFT \$              | 文字列の左からN番目の文字列を与える。                    | 94  |
| LEN \$(S)            | 文字列の長さを与える。                            | 93  |
| LET                  | 代入する。                                  | 67  |
| LIMIT                | メモリの上限を決める。                            | 168 |
| LINE                 | 線を引く。                                  | 121 |
| LIST                 | プログラム・リストを表示する。                        | 56  |
| LLIST                | プログラム・リストをプリンタ用紙に書く。                   | 55  |
| LOAD                 | カセット・テープに入ったプログラムをメモリに入れる。<br>(ロードする。) | 54  |
| LOADV                | グラフィック画面をロードする。                        | 154 |
| LPRINT               | プリンタ用紙に書きこむ。                           | 55  |

|                        |                               |     |
|------------------------|-------------------------------|-----|
| MAG                    | コメントで動かすキャラクタの大きさを指定する        | 148 |
| MID \$ (S, X, Y)       | 文字列Sの左からX番目から長さYの文字列を与える      | 94  |
| NEW                    | プログラムを消す。(リニアする。)             | 47  |
| ON ~ GOTO / ON ~ GOSUB | 分岐する行番号を選択する。                 | 74  |
| OUT                    | 出力ポートに出力する。                   | 170 |
| PAINT                  | 隠された動図をのりつぶす。                 | 133 |
| PEEK                   | メモリX番地の内容を与える。                | 169 |
| PLAY                   | 音楽を演奏する。                      | 97  |
| PLEN                   | 演奏 (PLAY の実行) 終了をチェックする。      | 100 |
| POKE                   | メモリへ書き込みをする。                  | 169 |
| PRINT                  | 画面に表示する。                      | 38  |
| PSET                   | 点を打つ。                         | 129 |
| PATTERN                | ・キャラクタをつくる。・文字などのパターンを変更する。   | 138 |
| READ                   | DATA 文のデータを読みとる。              | 79  |
| REM                    | コメントをつくる。                     | 60  |
| RESTORE                | READ 文により読み込む DATA 文の場所を指定する。 | 82  |
| RETURN                 | GOSUB で分岐したサブルーチンからもどる。       | 76  |
| RIGHT \$               | 文字列の右からX番目までを与える。             | 94  |
| RND (X)                | 乱数をつくる。                       | 137 |
| RUN                    | プログラムを実行する。                   | 47  |

|          |                                   |     |
|----------|-----------------------------------|-----|
| SAVE     | データのデータをカセットテープに保存する。             | 138 |
| SAVEV    | 現在表示中の画面をカセットテープに保存する。            | 138 |
| SCDIN    | サウンドカードのサウンドをONにする。               | 140 |
| SCREEN   | スクリーン画面、もしくはグラフィック画面に指定する。        | 140 |
| SGN(X)   | Xの正負符号を与える。                       | 141 |
| SOUND    | 効果音を発生させる。                        | 141 |
| SPC(X)   | スペースをあげる。                         | 142 |
| SPRITE   | グラフィック画面としてキャラクターをここに置く。          | 143 |
| STICK(n) | スティック(n)の方向を示す。                   | 145 |
| STOP     | プログラムを一時中断する。                     | 146 |
| STR\$(X) | 数値Xを文字列に変換する。                     | 147 |
| STRIG    | ジョイスティックのトリガーボタンの状態を示す。           | 148 |
| SYSTEM   | ホームページ画面にもとる。                     | 149 |
| TAB      | 表示位置の指定 (PRINT文で使用)               | 152 |
| TIME\$(  | 時刻を与える。                           | 153 |
| VAL\$(   | 文字列を数値に変換する。                      | 154 |
| VERIFY   | メモリのプログラムと、カセットテープに録音されたプログラムの比較。 | 154 |
| VERIFYV  | グラフィック画面から、ロードされたかを確認する。          | 154 |
| VPEEK    | 指定に書き込まれた内容を見せる。                  | 155 |
| VPOKE    | 画面に書き込みをする。                       | 155 |



REGA 車=40000円、2台設置

考工记 卷之四 凡一器者必象一器

8-11-2009

編著者佐藤正太郎 責任者サトウハチロー

00-742-7050

● 144 成功商人成功秘訣 (1) 社會人 500

2017年9月16日 星期六

國際・獨立・國際・國際

◆ 140 臺灣學術期刊網刊 | JH 99-4 110

© SGA 1999

無形社債発行を翌年  
終了。前年末はお取引と致しまして

100

株式会社 **セガ**<sup>TM</sup>・エンタープライゼス